5-2018

Contractive Autoencoding for Hierarchical Temporal Memory and Sparse Distributed Representation Binding

Luke G. Boudreau
lgb9267@rit.edu

Contractive Autoencoding for Hierarchical Temporal Memory and Sparse Distributed Representation Binding

Luke G. Boudreau

Contractive Autoencoding for Hierarchical Temporal Memory and Sparse Distributed Representation Binding

Luke G. Boudreau

May 2018

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

R·I·T | Kate Gleason
College of ENGINEERING

Department of Computer Engineering

Contractive Autoencoding for Hierarchical Temporal Memory and Sparse Distributed Representation Binding

Luke G. Boudreau

Committee Approval:

Dr. Dhireesha Kudithipudi *Advisor* Date
Professor of Computer Engineering

Dr. Ernest Fokoué Date
Associate Professor of School of Mathematical Sciences

Dr. Raymond Ptucha Date
Assistant Professor of Computer Engineering

Acknowledgments

There are any individuals that I would like to acknowledge. My parents, Karen and Jerry Boudreau, who have funded a significant portion of my education at RIT. My advisor, Dr. Kudithipudi, who has a thorough understanding of graduate student's priorities. Felipe Petroski Such, who has shown that when you find your passion you will pour all your energy in. Cody Tinker, who had looked beyond my idiosyncrasies, and continued to support me throughout my thesis. Eric Franz, for showing me how to open doors. Dillon Graham, for insightful and intellectual discussions regarding vector symbolic architectures, and without which a significant portion of this research would not exist. Michael Foster and Amanda Hartung, both of which had helped explain fundamental concepts related to Hierarchical Temporal Memory. James Mnatzaganian, for his advice regarding the continuation of Hierarchical Temporal Memory research. My Coach and his mantra of *don't think, just do*, which was a simple solution to what appeared to be difficult problems. The Rochester Institute of Technology, for providing me with all resources and opportunities.

I dedicate this thesis to my parents, Karen and Jerry Boudreau. Regardless of how far away or alone I am, my Mother's love has always provided that familiar feeling. My Father has been an exceptional role model in my life. His experience, perseverance, and discipline were crucial motivators for my study this field. Thank you both.

Abstract

Hierarchical Temporal Memory is a brain inspired memory prediction framework modeled after the uniform structure and connectivity of pyramidal neurons found in the human neocortex. Similar to the neocortex, Hierarchical Temporal Memory processes spatiotemporal information for anomaly detection and prediction. A critical component in the Hierarchical Temporal Memory algorithm is the Spatial Pooler, which is responsible for processing feedforward data into sparse distributed representations.

This study addresses three fundamental research questions for Hierarchical Temporal Memory algorithms. What are the metrics for understanding the semantic content of sparse distributed representations? The semantic content and relationships between representations was visualized with uniqueness matrices and dimensionality reduction techniques. How can spatial semantic information in images be encoded into binary representations for the Hierarchical Temporal Memory's Spatial Pooler? A Contractive Autoencoder was exploited to create binary representations containing spatial information from image data. The uniqueness matrix shows that the Contractive Autoencoder encodes spatial information with strong spatial semantic relationships. The final question is how can vector operations of sparse distributed representations be enhanced to produce separable representations? A binding operation that results in a novel vector was implemented as a circular bit shift between two binary vectors. Binding of labeled sparse distributed representations was shown to create separable representations, but more robust representations are limited by vector density.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Creating artificial general intelligence is a long sought goal for science, but today there still are no machine learning systems that come close to the capabilities of the human brain. The human brain is currently unrivaled in its ability to learn either mundane or sophisticated tasks and has a remarkable ability to adapt to new environments and conditions. Therefore, the human brain has been and will continue to be a source of inspiration for artificial general intelligence development. The brain is composed of multiple regions that are strongly integrated with each other and in some cases share overlapping functionality. Each region is made up of a significant number of cells called neurons; the average adult human neocortex alone contains 19 billion neurons and around 332 trillion synapses connecting those neurons [2]. Some research assumes that artificial general intelligence can be developed replicating the configuration of neurons and synapses in the human brain. However, there is still not enough of a neuroscientific understanding of the entire brain for this approach to be effective, and the intrusive nature of gathering detailed empirical data makes it difficult to get a complete overview of the physical structure and connections in the brain. In addition there are significant architectural limitations associated with the design of brain inspired silicon chips for machine learning applications. A simple neuron model can be represented in terms of a few transistors, yet it is unclear what the adequate abstract model of a particular neuron needs to be to achieve desired functionality.

The most recent generation of general purpose processors contain around 10 billion transistors, so a complete modeling of the neocortex alone would not be possible with simplistic neuron models. Although leading edge neuromorphic hardware like IBM's TrueNorth can model 1 million neurons with 5.4 billion transistors [3], the configurations of neurons and synapses remain a significant problem for establishing cognitive architectures.

A complete replication of the brain poses challenges for neuroscience with regard to: abstraction level, computer hardware, speed of computation, and verification of the complete system. In order to overcome these challenges, it is more effective to analyze specific brain regions. While the entire brain is composed of interconnected neurons and has considerable overlap in terms of functionality, it is known that certain regions of the brain are more responsible for high level cognition than others. Studying the brain by region reduces the scope of the problem, and provides a more scalable approach to modeling intelligent systems.

One region of interest in the human brain is the neocortex, which is associated with higher level intelligent behavior such as spatial reasoning, motor control, sensory perception. The neocortex is a common region in the mammalian brain, and in the human brain the neocortex is also responsible for language. The leading computational model of the neocortex is a theory known as *Hierarchical Temporal Memory*, which establishes a memory-prediction framework that can be used for prediction, anomaly detection, classification and sensorimotor applications. Hierarchical Temporal Memory (HTM) is a top down model in that it applies an overarching theory to the structure and function of the neocortex [4]. HTM has been used to successfully learn time-based sequences by using a composition of robust sparse distributed code of cellular activations [5]. This makes HTM an effective model of sequence memory in the neocortex. A product of this memory-prediction framework is the detection of new or unpredicted data, which can be utilized for anomaly detection applications.

Robust sequence memory has helped solved anomaly detection problems, and adapts to learn new patterns in streaming data [6]. The success of this model is dependent on good data representations within the algorithm. The internal data representation in HTM is known as a *Sparse Distributed Representation*, which contains semantic representations of the input data. In HTM theory the Spatial Pooler is the algorithm responsible for learning Sparse Distributed Representations of input data.

This research addresses three fundamental questions concerning HTM that focus on the Spatial Pooler. ***1. What are the metrics for measuring the quality of semantic content for Sparse Distributed Representations?*** Identifying appropriate criteria will help establish the correctness of data encoding techniques, and visualize the relationships in the representational geometry of various datasets. This will help HTM researchers analyze the content of Sparse Distributed Representations without assuming or focusing on producing separable representations that are sought in classification tasks.

2. How can spatial semantic information in images be encoded into binary representations for the Hierarchical Temporal Memory's Spatial Pooler? Image data contains many features such as object locations, orientation, shading, lighting, shadows, edges, and color. The relationships between these feature are extracted and untangled in the visual cortex for high level understanding. In HTM it is unclear what is the ideal level of features that should be passed to the Spatial Pooler, and how they should be encoded to preserve the important relationships among the image for the desired application. The added constraint for this problem is that these features and their semantic relationships must be encoded in a binary format, so regardless of how abstract or specific the feature space is it must be represented in a binary format for compatibility with the Spatial Pooler algorithm. This work examines the effect of the encoder on the semantic relationships found in the Sparse Distributed Representations.

3. How can vector operations of Sparse Distributed Representations be enhanced to produce separable representations? There are neuroscientific theories that suggest interaction between the hippocampus and neocortex for binding of representations. Vector Symbolic Architectures make use of a binding operation to create robust and complex representational structures. An implementation of a binding operation is explored for HTM's Sparse Distributed Representations, and the effect of the binding operation on the representational geometry is illustrated with dimensionality reduction techniques. In addition this work demonstrates the effectiveness of binding and superimposing Sparse Distributed Representations to create more robust representations. A small investigation will also explore the density increase of the vectors after superposition, which has a significant impact for manipulation and usefulness of these joined operations.

Chapter 2

Background

This research begins by looking at the brain as a source of inspiration for intelligence. A crucial area of interest in the brain is the mammalian neocortex because it responsible for many high level brain functions such as sensory perception, cognition, motor commands, spatial reasoning, and language. The neocortex is considered crucial to higher level cognition, and understanding how the neocortex gives rise to high level cognition is done by examining how it is constructed. Neuroscientists have discovered a uniform arrangement of pyramidal neurons in the neocortex. The pyramidal neuron's arrangement and intrinsic connectivity is found in six stacked layers that is commonly referred to as a *cortical column*. The neocortex is a sheet of cortical tissue that is composed of adjacent and repeated cortical columns. All regions in the neocortex function on the same principles regardless of the function of the region. The regions for vision, hearing, touch, and language are composed of the same repeated cortical columns [4]. The neocortex is flat by nature for some mammals such as rodents, but in humans the neocortex is much larger and highly folded onto itself. The same fundamental cortical column exists in different mammals' neocortex, so the use of the fundamental cortical column is likely responsible for high level cognition.

The modeling of the cortical column begins at the cellular level, and requires a more robust or biologically accurate neuron model than other artificial neurons. Traditional artificial neurons typically model one source of integration, whereas the

pyramidal neurons found in the neocortex have three sources of non-linear integration. Hierarchical Temporal Memory models these additional sources of information, and models the connectivity of pyramidal neurons in the cortical column. The connections of the artificial and neocortical pyramidal neurons are shown in Figure 2.1. Hierarchical Temporal Memory (HTM) proposes a theory or model of how information is processed in the neocortex with a focus on explaining how the cortical column functions. The repetition of the cortical column and the uniformity of the neocortex is one of the guiding principles for HTM theory, and the understanding of a cortical column will likely give insight into how the entire neocortex operates regardless of the information domain. This theory requires a more biologically accurate neuron model then common artificial neurons. Because of this HTM is a biologically influenced and constrained machine intelligence algorithm. A major challenge of HTM theory is the incorporation of new neuroscience research, which results in changes and enhancements to the algorithm to ensure its accuracy in modeling the cortical column.

2.1 HTM Algorithm Overview

The biological constraints in HTM are based on the uniformity and repetition of pyramidal neurons found in layers 2/3 of the neocortex. In HTM theory pyramidal neurons are referred to as the HTM neuron or *cells*. Cells are stacked into *column* in layers 2/3, and a collection of these columns form an HTM *region*. The region models the structure and functionality of the cortical column in the neocortex. The cells in a region receive information from three different sources; feedforward, contextual, and feedback information is transmitted through three different connections as shown in Figure 2.1. The connections to cells are referred to as *segments*, and each segment is composed of a set of *synapses*. Information flows to cells by means of synapses, and in HTM information is represented with binary values. A cell has three possible states:

active, inactive, or predicted. The state of each cell is induced by the information it receives. The processing of feedforward input is done by algorithm called the *SpatialPooler*, and the processing of feedforward and contextual input is done by algorithm called Temporal Memory. The feedback connections are not addressed because current HTM theory suggests that feedback is an optional component [7].

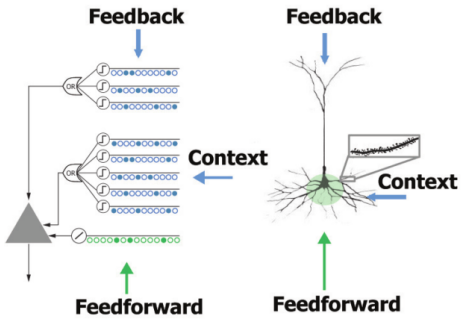


Figure 2.1: The HTM Cell on the left, and a pyramidal neuron on the right. Both have feedforward, context, and feedback connections [1].

2.1.1 Spatial Pooler Algorithm

The Spatial Pooler is an unsupervised machine learning algorithm that converts binary input into a binary *Sparse Distributed Representation*. The purpose of the Spatial Pooler is to continually encode streams of binary sensory data into Sparse Distributed Representations, while at the same time ensuring that a generalized representation is produced for similar inputs. A Sparse Distributed Representation (SDR) is generated by processing the feedforward input to cells in the HTM region.

In the Spatial Pooler there is no connectivity between cells; each cell only receives information from *proximal segments*. Inter-cellular connectivity is described in section Section 2.1.4, when the Spatial Pooler is combined with other components of HTM theory. The proximal segments transmit the same information to all cells in a particular column. The proximal segment contains a set of potential proximal synapses, and each proximal synapses has a weight or *permanence value*, ϕ, that de-

termines if the synapses is connected. A proximal synapse is considered "potential connections" because its connectivity depends on the permanence value. Permanence values greater than some threshold θ will result in connected synapses, but values less than θ will be unconnected. The typical value used for the threshold of connectivity is 0.5, so only permanence values greater than or equal to 0.5 will result in a connected synapse. The synapses are learned connections, and is inevitable that throughout learning some synapses will be connected and some will be disconnected based on the permanence value.

The initialization of the Spatial Pooler is done by establishing the all the column's proximal segment's connectivity to the input space. Each column's proximal segment has potential connections to only a fraction of the input space. Each column in the region has a receptive field, which is a subregion of the entire input space. The synapses in the column's segment connect to only a fraction of the possible inputs in the receptive field. The organization of the column's receptive fields to the input space is known as *topology*. An example receptive field is shown in Figure 2.2. Topology is useful when there is some natural ordering or spatial relationship among the input space [1]. A possible method to implement topology is to have neighboring column's receptive fields overlap, which gives the opportunity for neighboring columns to potentially receive the same input. The lack of topology indicates that the receptive fields are global with respect to the input space. In this case the receptive field for each column is the entire input space, and the proximal synapses can potentially be connected to any input variable. With or without topology, the permanence values of each synapse in a proximal segment is randomly initialized. This leads to synapses that are both connected and disconnected to the input space.

After Initialization, the Spatial Pooler can process and learn from input data. Cells in the Spatial Pooler become active when there is enough input to the connected synapses on the proximal segment. Because all cells within a column use the

same proximal segment, in the Spatial Pooler all cells become active together. However, there are scenarios where one cell or only a portion of cells within a column become active due to feedforawrd input, and scenarios where cells enter the depolarized or predicted state. These phenomena occur when the Spatial Pooler is used other components of HTM theory, which is explained in Section 2.1.4. The remainder of this section will consider that the cells within column share the same state, and the possibility of these states are either *active* or *inactive*. The presence of any active cells in a column indicate that the column is active, i.e. a column is considered active if there are any active cells within the column.

Columns of cells with proximal segments are shown in Figure 2.2, this diagram illustrates the structure of the Spatial Pooler with feedforward connections. The level of column activation is known as the *overlap* value, and if the value is below a certain threshold θ_{stim} then the column has an overlap value of 0. Overlap values are inhibited in a global or local manner to determine which columns (and consequently the cells within) remain active. Inhibition is used to introduce sparseness in the resulting active column population. Global inhibition sets top k column's with the largest overlap values to active. Local inhibition establishes neighborhoods of columns, and selects columns with the top $k\%$ overlaps to be active within their neighborhoods. In either case the targeted column activation density is strictly 2% for global inhibition, and it's approximately 2% for local inhibition [1]. When topology is implemented in the Spatial Pooler, then local inhibition takes into consideration the spatial relationships among the input space.

After inhibition, the output of the Spatial Pooler is a set of columns with active cells. However, the active columns can be represented as a binary vector or SDR, this is the learned representation of the input. An example SDR is shown set of column/cell activations can be represented by a binary vector as shown in Figure 2.6. The final part of the Spatial Pooler algorithm is to adjust the permanence values of the

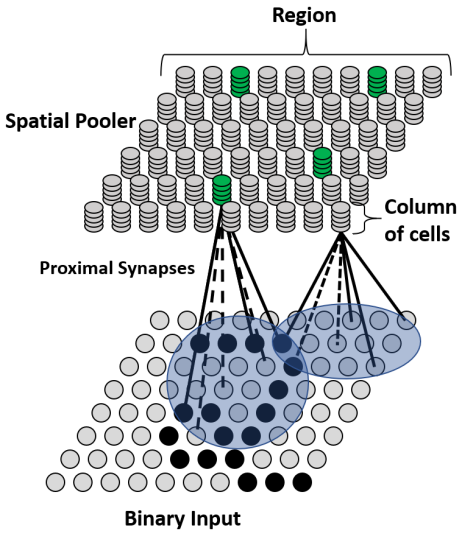


Figure 2.2: An example region for the Spatial Pooler. Two columns are shown with their proximal segments. Each proximal segment contains a set of proximal synapses (dotted and solid lines). Both columns have different receptive fields (blue circles). Each proximal segment has six potential synapses; connected and disconnected synapses are represented by solid and dotted lines respectively.

synapses based on the surviving active columns. Only proximal synapses connected to active columns are updated by a Hebbian learning mechanism: the permanence values of synapses that have an active input are strengthened, but the permanence value of synapses with inactive input are weakened.

The Spatial Pooler behavior can be described as a three step algorithm that involves an activation calculation for each column, a non-linear inhibition process, and a Hebbian learning step. The mathematical formalization of the Spatial Pooler algorithm presented in this work is attributed to [8]. Examples for all three steps in the Spatial Pooler algorithm are illustrated: Figure 2.3 shows overlap step, Figure 2.4 shows the inhibition step, and Figure 2.5 shows the learning step.

1. **Overlap** This stage determines a set of active cells/columns given an input sample. A column is considered active if there are enough active and connected synapses. A connected synapse is one where its permanence value is greater than some threshold, ρ_s, which is determined in (2.1). The permanence values

are represented by $\Phi \in [0,1]^{mxq}$, where m is the number of columns in the region and q is the number of proximal synapses per column. The input space is represented by X^{mxq}. An active synapse is one where the input to the synapse is a binary value of '1'. The column overlap is the sum of all the active and connected synapses belonging to the column's proximal segment, and can be computed using the dot product or matrix multiplication (2.2). The index $i \in [0, m)$ in (2.2) and (2.3) identify the specific columns. The calculation is performed for each column in the region denoted by (2.2), which produces a vector of overlap values for all the columns $\vec{\alpha} \in \mathbb{Z}^{1xm}$. If the column overlap value is larger than the proximal segment threshold, ρ_d, then the column is active otherwise it is inactive (2.3). In some implementations of the Spatial Pooler, a mechanism called *boosting* is used to encourage columns with few historical activations to become active. This value is represented by an overlap coefficient $\vec{b_i}$ in 2.3; different methods for the calculation of a column's boost value are explored in [8][1]. Figure 2.3 shows the overlap computation of two columns with different proximal segments. The result of this stage is a set of all active column overlap values, and is defined as $\vec{\alpha} \in \mathbb{Z}^{1xm}$.

$$Y \equiv I(\Phi \geq \rho_s) \tag{2.1}$$

$$\vec{\hat{\alpha}_i} \equiv X_i \bullet Y_i \tag{2.2}$$

$$\vec{\alpha} \equiv \begin{cases} \vec{\hat{\alpha}_i}\vec{b_i} & \vec{\hat{\alpha}_i} \geq \rho_d, \\ 0 & \text{else} \end{cases} \forall i \tag{2.3}$$

2. **Inhibition** The inhibition stage introduces a level sparsity in the number of

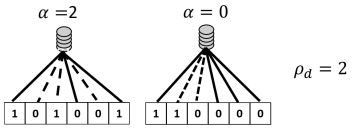


Figure 2.3: The proximal segment threshold for this example is $\rho_d = 2$, and the connected proximal synapse value is $\rho_s = 0.5$. Connected proximal synapses are solid black lines ($\phi \geq \rho_s$), and disconnected proximal synapses are dotted lines ($\phi < \rho_s$). The overlap value for the left column is three, and the overlap for the right column is zero according to equation (2.3).

active columns that will form the SDR. The two types of inhibition are global and local inhibition. In either method a column's neighbors are determined by its neighborhood mask, H_i, which is an element-wise multiplication with the set of active columns from the previous step (2.4). In the case of global inhibition the neighborhood mask is the entire region (all columns within the region), so each column effectively uses the same neighborhood mask. In (2.4) ρ_c determines the level of sparsity, and is typically defined around 2% of the total number of columns in the region [9][8][1]. The $kmax(x, k)$ operation picks the k^{th} largest value from x. The k$^{\text{th}}$ max overlap value for the specific column i is known as γ_i, which is calculated in (2.4). The k-max values, γ_i are used to inhibit the set of active columns, $\vec{\alpha}$, based on the overlap values (2.5). The result of this step is a set of active columns or a sparse distributed representation, which is represented by the binary vector $\vec{c}$.

$$\vec{\gamma_i} \equiv max(kmax(H_i \odot \vec{\alpha}, \rho_c), 1) \forall i \tag{2.4}$$

$$\vec{\hat{c}} \equiv I(\vec{\alpha_i} \geq \vec{\gamma_i}) \forall i \tag{2.5}$$

3. **Learning** The final step is a form of Hebbian learning. Learning is only performed on the synapses of the columns that remained active after the inhibi-

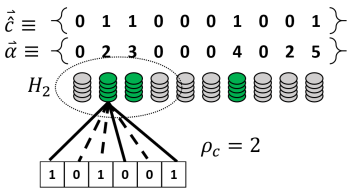


Figure 2.4: An example of local inhibition for the second column ($i = 2$), where the neighborhood mask, H_2, for the second column consists of the four columns in the dotted circle. γ_2 is set to the second largest overlap value in the neighborhood because $\rho_c = 2$ according to (2.4). The second column is considered active because it overlap value, $\alpha_2 = 2$ is greater than or equal to $\gamma_2 = 2$.

tion step. The synapse permanence values are scalar weights between 0 and 1. The synapses with active inputs have their permanence values increased, and synapses with inactive inputs have their permanence decreased. The update values for all active columns are determined by equation (2.6). Inactive column's proximal synapses remain unaffected. The learning rate is controlled by the parameters ϕ_+ and ϕ_- (increment and decrement values). The application of the update is shown in equation (2.7), and ensures that the synapse permanence values stay between 0 and 1.

$$\delta\Phi \equiv \vec{\tilde{c}}^T \odot (\phi_+ X - (\phi_- \bar{X})) \tag{2.6}$$

$$\Phi \equiv clip(\Phi \oplus \delta\Phi, 0, 1) \tag{2.7}$$

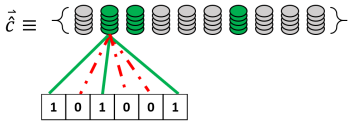


Figure 2.5: Learning is performed on active columns only which are represented in green. A Hebbian learning principles governs synapse permanence value update. The green lines illustrate the synapse permanence values that are increased by ϕ_+, and the red dotted lines illustrate that synapse permanence values are decreased by ϕ_-.

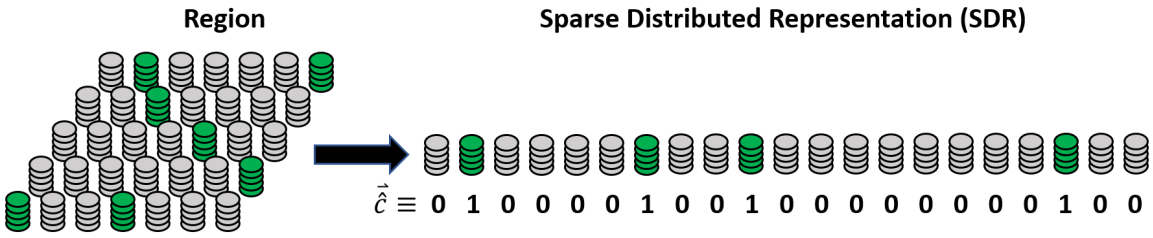


Figure 2.6: The representation of the active columns after inhibition is represented by a binary vector. The active and inactive columns are represented in an SDR with '1' and '0' respectively.

2.1.2 Spatial Pooler Performance Metrics

There are four metrics for evaluating Spatial Pooler performance based on the inputs and outputs [1]. These metrics can help establish when the Spatial Pooler has finished learning from the given data and rate of learning for the Spatial Pooler. It also provides information about column utilization and sparseness, which is used to evaluate the robustness of the SDRs.

1. **Sparseness** The sparseness of the Spatial Pooler is the percentage of active columns at a particular time step. This metric can be used to determine the sparsity of the inputs to the Spatial Pooler as well as the number of columns utilized. This metric is typically used to observe the sparsity of columns when local inhibition is utilized.

2. **Entropy** The entropy is the average activation frequency of each mini-column, and determines if the Spatial Pooler is actively using every column efficiently. This metric determines if all the columns in the region are being utilized equally, so the entropy effectively evaluates the distributed degree of the representation.

3. **Noise Robustness** The measurement of sensitivity to random bit flips ensures that the Spatial Pooler is resilient to noise. The more noise required in the input to substantially change the output SDR, the more robust the representations are to noise.

4. **Stability** Stability ensures that the active columns remain consistent when there is no changes to the input stream. This metric measures the degree to which the Spatial Pooler is learning; a low stability value means that the Spatial Pooler is actively learning from new samples, and a high stability value can be interpreted that the Spatial Pooler has not learned new spatial patterns since the last stability measurement.

2.1.3 Properties of Sparse Distributed Representations

It was shown that the active columns in an HTM region can be represented by a Sparse Distributed Representation as shown in Figure 2.6. The benefits for representing data in this as a binary sparse distributed representation is robustness to noise, effective measure of similarity between representations, and bitwise operations for manipulation.

The distributed nature of the representation ensure that the semantic meaning is *distributed* over the entire vector, for a single element in a distributed representation does not have significant meaning related to the whole representation. This creates robustness if a bit is not encoded correctly or is prone to noise, but most of this resiliency is only found with high dimensional SDRs greater than or equal to 2048 bits or columns [10]. As a consequence the dimensionality governs the total number of columns and cells within an HTM region. The use of an HTM region should be sufficiently large to gain the robustness benefits. The sparse nature of the representations is imposed from the Spatial Pooler inhibition process. The typical targeted values of sparseness are around 2%, which results in about 2% of the bits in the SDR are active depending on the type of inhibition used [1].

The distributed representation found in SDRs is contrasted with a *localist* representation; a single computing element represents a single entity in a local representation, but a single computing element in a distributed representation represents many

different entities. As with any representation method, there are trade-offs associated with using distributed representations over local representations. Geoffrey Hinton claims that in general distributed representations are adequate for content addressable memory, automatic generalization, and the selection of the rule that best fits the current situation [11]. This claim can be summarized in that distributed representations are more adequate for models of memory structure and processing in the brain.

Because the SDR is a binary vector it is trivial to calculate the degree of similarity between two SDRs, which can be computed with a sum of logical AND operations. However, it can also be computed by taking the dot product between two SDR vectors. Numenta calls this measure of similarity *SDR overlap* [10]. Computing similarity or dissimilarity is important when combined with the union property of SDRs. The union property allows a set of SDRs to be composed into a single SDR or union of SDRs. The union SDR contains an unordered set of SDRs that were superimposed together by taking the logical OR operation between all SDR vectors. The determination of set membership is done by computing the SDR overlap (dot product) between a the union SDR and a known SDR, and if the value of the similarity is above some threshold, θ, then that known SDR is considered a part of the set. There is a limit to how many vectors can be stored in a set because as the number of vectors in the union increases the false positive rate for membership increases as well [10].

In context of HTM the SDRs represent the active columns in the Spatial Pooler, and more specifically the SDR represents the semantic information and relationships in the feedforward input. However, in the broader context of HTM research and the neocortex an SDR is considered to be the state of any cortical neuron population. In the Temporal Memory portion of HTM as described in Section 2.1.4, the set of active cells and predicted cells are each considered SDRs. The properties described in this section are still applicable for this definition of SDRs. For the purposes of this work,

the use of *SDR* will refer to the set of active columns that is generated by the Spatial Pooler.

2.1.4 Temporal Memory Algorithm

The processing of feedforward and contextual input for pyramidal neurons in the neocortex is modeled by the *Temporal Memory* algorithm in HTM theory. The Spatial Pooler is always utilized with Temporal Memory, but there modifications to the Spatial Pooler algorithm. The input to the Spatial Pooler is now treated as spatio-temporal data, so the ordering of spatial input has contextual meaning. The contextual data is stored at the cellular level; different cell activations within the same column in a region represent the same spatial data in different contexts. The contextual data shown in Figure 2.1 received from other cells within the Region. The transmission of contextual data between cells is done through *distal segments*, where a distal segment is activated through cell activations. Each cell has several distal segments, and if the cell receives enough activity on any of its distal segments, the cell enters the predicted state. The possible cell states for Temporal Memory and the proximal and distal synapses are shown in Figure 2.7.

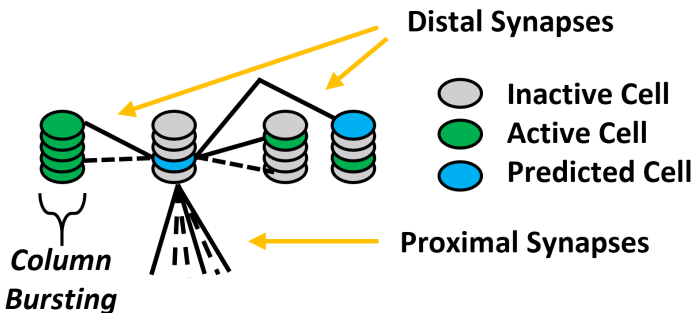


Figure 2.7: Each cell can be in one of three states: deactivated (grey), active (green), predicted/depolarized (blue).

All cells in the same column correspond to the same spatial data, but different contexts of that spatial data trigger different cell activations within a particular col-

umn. A cell is a representation of some spatial data in a specific context. If multiple cells within a column are in the predicted state, then there are multiple spatial predictions or possible subsequences. If there is no prediction of spatial data and the Spatial Pooler algorithm selects are particular column for activation, then a phenomena called *busting* occurs. The column bursts by activating all cells within the column. The cell with the least distal segments within the column is selected to establish a new distal segment. The connectivity of the distal synapses in the new distal segment is determined by the cells that were activated in the previous time step. This is done to learn the new temporal patterns in the spatial data. Because HTM is a continuous online learning system, it adapts to changes in the data it processes.

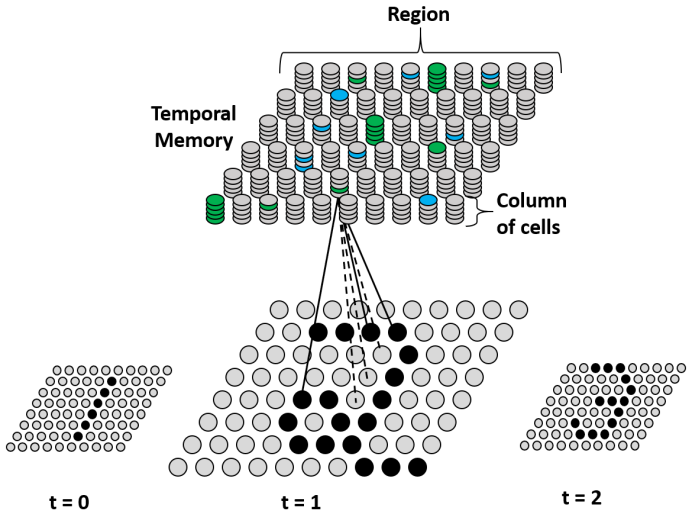


Figure 2.8: The HTM Region with active and predicted cells for Temporal Memory. Distal segments and synapses are omitted. The region learns to predict changes in input data.

The formalization of sequence memory was defined in [5], and is annotated here for completeness. The set of active cells for a given time step is stored in A^t which is indexed through i cells and j columns. The calculation of active cells, A^t, requires the active columns that were selected by the Spatial Pooler. The W^t vector represents the indices of the active columns or bits in SDR produced by the Spatial Pooler algorithm ($\vec{c}$ in (2.5)). For this work it can be assumed that $W^t \equiv \vec{c}$. Only the predicted cells

are selected to be active within an active column, which is described by the first case in (2.8). If there are no predicted cells within the column then the column bursts. All cells within the bursted column are put into the active state, which is described by the second case in (2.8).

$$a_{ij} = \begin{cases} 1 \ \textit{if} \ j \in W^t \ \textit{and} \ \pi_{ij}^{t-1} = 1 \\ 1 \ \textit{if} \ j \in W^t \ \textit{and} \ \sum_i \pi_{ij}^{t-1} = 0 \\ 0 \ \textit{else} \end{cases} \tag{2.8}$$

The set of predicted cells for an input is given by Π^t where π_{ij} is the i^{th} cell of the j^{th} column. Cells are predicted if there is any segment that has an activation value greater than some threshold θ (2.9). The predicted cells, Π^t, are determined based on the current active cells, A^t, calculated in (2.8). A set of distal synapses form a distal segment, and each cell can have many distal segments. The distal segments and synapses represented by a D_{ij}^d matrix where the weight is stored in the d^{th} segment's connection to the i^{th} cell of the j^{th} column. Every cell has a distinct D_{ij}^d matrix with values $[0, 1]$. When the D_{ij}^d is used in (2.9) the synapse permanences are rounded to binary values, which is denoted by the $\tilde{D}_{ij}^d$ matrix.

$$\pi_{ij} = \begin{cases} 1 \ \textit{if} \ \exists_d \|\tilde{D}_{ij}^d \circ A^t\|_1 > \theta \\ 0 \ \textit{else} \end{cases} \tag{2.9}$$

The learning or adjustment of the synapse permanence values only occur on segments that correctly predicted cells from the previous time step (2.10). These segments are those that caused the prediction of cells that were subsequently active in the next time step.

$$\forall_{j \in W^t}(\pi i j^{t-1}) \ \textit{and} \ \|\tilde{D}_{ij}^d \circ A^{t-1}\|_1 > \theta \tag{2.10}$$

When the HTM region is first learning sequences there will be no cells in the predicted state, and columns will burst until cells are predicted. In this early learning phase the adjustment of synapse permanence values will be done on the most active segment across all cells for the winning columns of W^t (2.11). The most active segment is determined by computing the dot product between the synapse permanence values and the previously active cells. Because there was not enough activity or the synapses were unconnected, the weights are mapped to binary values $\dot{D}^d_{ij}$. The binary values are calculated based on the nonzero and zero entries in D^d_{ij}, the segment that produces the largest cell activation is the candidate for learning.

$$\forall_{j \in W^t}(\sum_i \pi ij^{t-1} = 0) \ and \ \|\dot{D}^d_{ij} \circ A^{t-1}\|_1 = max_i(\|\dot{D}^d_{ij} \circ A^{t-1}\|_1) \tag{2.11}$$

The learning rule for selected segments in (2.11) is a form of Hebbian learning rule on synapses (2.12).

$$\Delta D^d_{ij} = p^+(\dot{D}^d_{ij} \circ A^{t-1}) - p^- \dot{D}^d_{ij} \tag{2.12}$$

There is a small decay applied to those distal segments that induced cell predictions where cells did not become active in the subsequent time step. These segments responsible for incorrect predictions have a small decrement across all permanence values (2.13).

$$\Delta D^d_{ij} = p^{--} \dot{D}^d_{ij} \ where \ a^t_{ij} = 0 \ and \ \|\tilde{D}^d_{ij} \circ A^{t-1}\|_1 > \theta \tag{2.13}$$

2.1.4.1 Initialization of distal segments and synapses

The initialization of the distal segments is not clear according to [5] as (2.11) does not explain how the D_{ij}^{d} connections are formed. In [6] a greedy approach was used to establish distal segments from the current activated cells, A^{t}, to the previous activated cells A^{t-1}. In this manner a history of previous cell activations was required to keep track of where to initialize distal segments.

2.1.5 Encoding Data for HTM

Data must be converted to a binary representation of pred in order to be used for input to the Spatial Pooler. In cases where data is not binary an encoder must be used to convert the data to an SDR. Numenta has proposed several types of encoders for encoding scalar data and spatial data [12]. After encoding, similar semantic input should have overlapping active bits in their SDRs after being encoded. The encoder has a strong influence on what aspects of the data contribute to the similarity.

Numenta has proposed a formalization for the encoding process in their encoding work [12]. This formalization considers an arbitrary space A and $S(n,k)$ an SDR with n total bits and k active bits where f is a function of $A \rightarrow S(n,k)$. In order to evaluate the encoder, a distance metrics over space A is established in equation (2.14). The performance of the encoder can be evaluated by comparing the distances scores of pairs of inputs with the overlaps of their encoding. Equation (2.15) shows that encodings with more overlapping bits have greater semantic similarity.

$$d_A : A \times A \rightarrow \mathbb{R} \begin{cases} \forall x, y \in A, d_A(x,y) \geq 0 \\ \forall x, y \in A, d_A(x,y) = d_A(y,x) \\ \forall x \in A, d_A(x,x) = 0 \end{cases} \tag{2.14}$$

$$O(f(w), f(x)) \geq O(f(y), f(z)) \Leftrightarrow d_A(w, x) \leq d_A(y, z) \tag{2.15}$$

2.1.6 Applications

The SDRs in HTM can be used for traditional classification problems, but the temporal memory is more strongly used for prediction and anomaly detection tasks. HTM does not currently explain how long term memory is recalled and stored, so an external classifier is required for practical application tasks. Common external classifiers used are SVM or softmax [8][1].

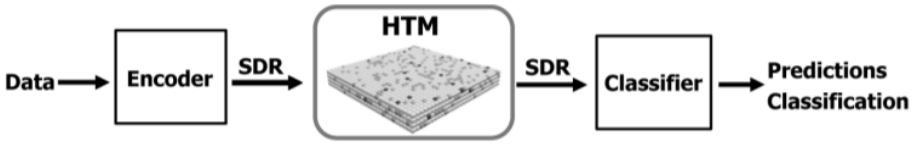


Figure 2.9: High level architecture for an HTM learning system. Because memory and recollection is not well understood in the neocortex, the SDRs must be passed to a classifier for classification of input. An SVM or Softmax layer are typically used for classification problems.

2.2 Vector Symbolic Architectures

Jackendoff proposed four challenges for cognitive neuroscience related to the understanding of grammer in language in 2002 because traditional connectionist models fail to address symbolic processing that is necessary for language [13][14]. Most importantly these challenges are *not* exclusive to language, but are central to higher level cognition. The main issue with connectionist models is the lack of symbolic structure based representations, and the construction of complex thoughts or sentences from basic thoughts or words [15]. Jackendoff's first problem for cognitive neuroscience is the *binding problem*, which applies to vision as well as language. An example of the binding problem in the visual domain is given by a cluttered scene of objects with

each object presenting different features/attributes such as a red square and a blue circle. In this scene, how does the brain associate red with square and blue with circle and not vice versa? These issues were presented in the form of challenges by Ray Jackendoff. In response, Ross Gaylor proposed a connection based model that answered Jackendoff's questions , and coined the term "Vector Symbolic Architecture" [14].

The central element in Vector Symbolic Architecture (VSA) is the vector representations, which can have different restrictions depending on the architecture. The difference in these representations leads to different implementations of the basic vector operations. The differences in VSA implementations are further described in section Section 2.2.4 alongside descriptions of the vector operations. There are common representations

The distributed vectors in VSA implementations are designed for powerful recursive binding operations that are necessary for the processing of simple and complex concepts [16]. In addition to variable creation, the binding operation can be used to create novel concepts and content addressable memory structures. Content addressable memory functions is a type of memory system where the the content of the representations is used to traverse through memory. This approach to memory design is similar to how memory in the brain works, and is unlike the memory paradigm that has driven classical computation [11]. The binding problem has been posed in other contexts, and it has been suggested that binding in the brain is crucial for weaving temporal information into signals [17].

2.2.1 Binding

In the context of VSAs there are thematic relations or roles that are associated with some attribute or filler. The role and filler vectors are bound (or associated) together in such a way. The implementation of this operation must not increase the dimension-

ality of the resulting bound vector because in practice it will be difficult to establish an upper bound on memory and the time complexity on subsequent operations will grow exponentially. Therefore, binding vectors without increasing the dimensionality of representations ensures that all structures (either basic or complex) have a consistent time complexity for all operations, and establishes an upper memory limit for the storage of vectors. The composition of the bound vectors is unique relative to both composites [14]. The binding operation is invertible allowing for the recollection of either of the composing vectors given the other, which allows for the recollection of memory.

2.2.2 Superposition

More complex structures can be created by superimposing vectors into a single vector. The composition vectors could be bound attribute-value pairs of vectors, and the superposition would be a collection of these role-filler pairs. Unlike the binding operation, the superposition operation preserves the similarity between its elements.

2.2.3 Permutation

According to Kanerva, the permutation operation is versatile operation in for binary vectors [18]. An advanced use of the permutation operation is to implement a thinning technique for binary vectors [19].

2.2.4 VSA Implementations

There are various implementations of Vector Symbolic Architectures, and each implementation comes with advantages and disadvantages. These implementations differ in the choice of values used for the vector representations and size of vectors, which in turn lead to different implementations of vector operations. A common theme among all implementations is that they all address the combinatorial explosion by

providing a binding mechanism, but the method in which the outer product reduction occurs differs. The implementations listed here are not meant to be exhaustive, but to illustrate the different approaches to implementations of VSA theory.

All VSAs provide a solution for the representation of sequences/recursive/tree structures [20] associations of items may be the subject of other associations. The reduction must be reversible, so that expansion can be done in both directions [20].

2.2.4.1 MAP

Ross Gayler proposed Multiply, Addition, Permutation (MAP) as VSA symbolic architecture as a direct solution to Jackendoff's challenges. Gayler discusses how to generate vectors for novel concepts, which is crucial for adding to existing structures or analogical reasoning [21].

2.2.4.2 Holographic Reduced Representations

Tony Plate proposed using circular convolutions to construct associations of vectors without increasing the dimensionality of the bound vector. Since vectors are bound using circular convolution and are unbound using circular correlation, these memories are coined *holographic* [20].

2.2.4.3 Binary Spatter Codes & Hyperdimensional Computing

Pentii Kanerva's approach to Vector symbolic architectures is to create high-dimensional binary vectors, which were born out of a model for long term memory called Sparse Distributed Memory [22]. The operations here utilize bitwise operations for binding, superposition, and permutation.

2.3 Representation Learning

Data in this world has many forms and dimensionality, and the creation of efficient representations for data is crucial for the application of learning algorithms. In HTM theory there are simple encoders for low dimensional data, which creates a need to explore other methods of encoding data into robust representations. The continuous online learning nature of HTM systems conflicts with neural networks that extract features based on labeled data (Alexnet,VGG-16,Resnet), so instead unsupervised representation learning systems are more in line with HTM theory. Autoencoders have provided a way to learn hidden representations by learning to reconstruct input in an unsupervised manner (no class labels). It has been shown that initializing a deep neural network with a trained autoencoders weights on the same data dataset improves performance of the deep neural networks.

2.3.1 Autoencoders

The Autoencoder network is an unsupervised machine learning architecture that learns to form efficient codes for input data. A single autoencoder is composed of two layers: the encoding layer and decoding layer, or encoder and decoder. The encoding layer transforms the input into the encoded representation (hidden units), and the decoding layer transforms the encoded representation back into the input domain. Hidden unit activations are computed with the encoder, which is typically a linear combination of the input and the encoder's weights followed by a non-linear activation function (2.16). The decoding layer is a linear combination of the decoder's weights and the encoded representation (hidden units) (2.17).The learning of these encoded representations (or hidden unit activations) is guided by how well it is able to reconstruct the input data. Learning of the weights and biases is accomplished an optimization of a loss function, which typically takes the form of mean-squared

error. The reconstruction cost is defined as a squared error between the input and reconstructed input (2.18).

$$f_\theta(X) = h = \sigma(WX + b) \tag{2.16}$$

$$g_\theta(h) = \sigma(WX + b) \tag{2.17}$$

$$J(\theta) = \sum_t L(x, g_\theta(f)) \tag{2.18}$$

There are several variations of the autoencoder with many tunable hyperparameters. For example the lack of activation functions for both the encoder and decoder allow the network to learns a similar subspace as Principle Component Analysis [23]. However, the lack of non-linear activation functions in the autoencoder do not allow for layers to be stacked to create "deep" networks. Regularization of the weights can also be applied to the autoencoder to prevent over-fitting on the dataset, which becomes an issue when the number of hidden units approaches or exceeds the dimensionality of the input data. In many cases the l2 regularization term is applied to the autoencoder's cost function, but another form of regularization that is used are tied weights. Tying the weights of the encoder and decoder ($W^D = (W^E)^T$) means that the weights of the encoding layer and decoding layer are identical but transposed.

Traditional applications of the autoencoder include dimensionality reduction because the dimension of the encoded representation could be smaller than the input space dimensionality. Another usage scenario is to train the autoencoder on a labeled dataset, and use the weights of the autoencoder to initialize the weights of a multilayer perceptron network. This approach to initialization has helped improve the classification accuracy of these networks.

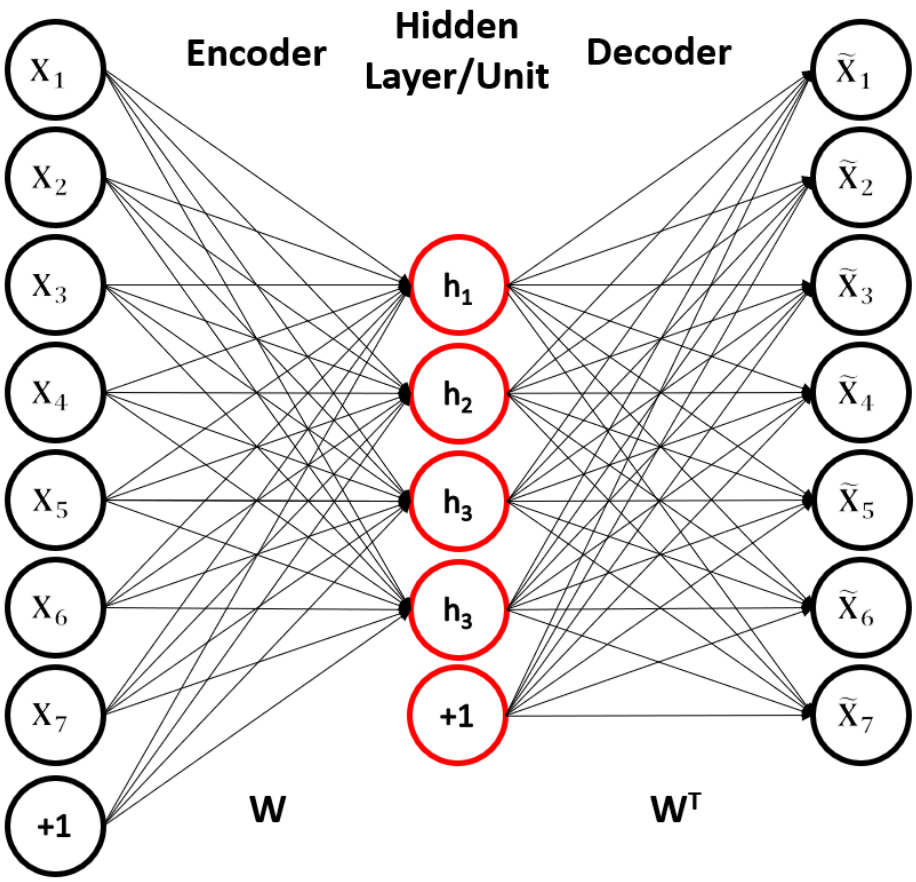


Figure 2.10: An under-complete autoencoder with bias terms. The reconstructed input is represented on the right.

2.3.2 Sparse Autoencoders

Another form of regularization that is applied to autoencoders is enforcing sparsity in the hidden unit activations; sparse regularization will drive many of hidden unit activation values to zero, and will enforce very few nonzero activation values. The regularization term is added to the reconstructed cost to create the overall cost function. Two commonly used sparse activation terms are the student-t and Kullback-Leibler divergence. Sparse autoencoders trained on natural image patches learn edges that are similar to the gabor filters in V1 area of the visual cortex [24].

2.3.3 The Contractive Autoencoder

The Contractive Autoencoder (CAE) was proposed in 2011 by Rifai et. al as way to learn robust representations [25]. This autoencoder implementation includes a regularization penalty on hidden unit activations. The regularization term is the Frobenius norm of the Jacobian weight matrix (2.19). The analytic penalty is minimized with low-valued first order derivatives, which leads to the notion of *flatness* or *contraction* as described by the authors. This penalty encourages training data to be lie on a relatively low-dimensional manifold in the high dimensional feature space by contracting the input data in the directions of small variations. These small variations in the input are captured by the autoencoder because it helps differentiate the reconstruction of neighboring training examples.

$$\|J_f(x)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(x)}{\partial x_i} \right)^2 = \sum_{i=1}^{d_h} (h_i(1-h_i))^2 \sum_{j=1}^{d_x} W_{ij}^2 \tag{2.19}$$

The encoder utilizes a non-linear activation (sigmoid), which constrains the hidden layer activations to values between zero and one. The decoder has no activation function (linear transformation), or in some cases a sigmoid activation function is used. The full autoencoder is shown in (2.20), which combines the encoder and decoder together for the reconstruction y of x (input). According to Rifai *et al*, tied weights are used in the CAE to avoid scaling and expansion in the encoder and decoder respectively. This ensures that the weights (learned features) allow the transformation to a flat manifold, but also provide the capability to reconstruct the input space. Tied weights also avoid degenerate solutions [23].

$$y = W^T \sigma(Wx + b_h) + b_y \tag{2.20}$$

The CAE minimizes its regularization term by ensuring that the derivative is small, which leads to saturated values in the case of a sigmoid activation function.

The derivative is near zero for a sigmoid function when the values are either close to zero or one. The non-saturated values show sensitivity to the input, but the degree of sensitivity is *infinitesimal*. This is later alleviated by introducing a penalty on the Hessian in addition to the Jacobian [26].

2.3.4 Multiple Layers

Multiple layers can be realized with autoencoders to learn more invariant relationships in the data. As long as non-linear activation functions exist between layers, then subsequent layers will learn non-linear transformations. However, the connectivity and loss functions become more complex as more layers are introduced. Subsequent layers can be implemented as a reconstruction of the previous layer's hidden units, or there can be several encoding layers followed by several decoding layers.

2.4 t-distributed stochastic neighbor embedding

Visualizing the structure and shape of high-dimensional data is an critical problem that exists in many domains, and there are several techniques for representing high-dimensional in lower dimensions. One of these methods is t-distributed stochastic neighbor embedding, which is a nonlinear dimensionality reduction technique that can reduce data to two or three dimensions [27]. The t-distributed stochastic neighbor embedding (t-SNE) algorithm is capable of capturing the local structure of data quite well. However, t-SNE is also prone to misinterpretations that arise with different values of hyperparameters [28].

Two of the main hyperparmeters are the number of steps in the algorithm and the *perplexity*. The perplexity is a smooth measure of the number of neighbors each point has, and should be smaller than the total number of points in the dataset. Additionally it is a measure of entropy in the system.

2.5 Memory in the Neocortex

Because of the invasive nature of measuring and difficulty recording neuron activity at the necessary granularity, it is difficult to study, locate, and isolate specific or individual memories in the brain. There are many theories of how stimulus is encoded in the brain, yet there is no major evidence that one of these encoding mechanisms drive all others. Population encoding finds meaning in a population of neurons in a defined region of the brain. A population of neurons can be modeled with a vector of values; each value in the vector represents the activation strength of a neuron. In the extreme case the values of the vector could be represented with binary values, which abstractly indicates if the neuron is active with no information regarding the strength of the activation. When looking at a several thousand neurons or a vector with thousands of elements this space becomes high-dimensional. The individual elements of these vectors do not have specific meaning, but the meaning of the vector as a whole contains information about where the vectors lies in the high-dimensional space.

While it is commonly accepted that there are many categories of memory in the brain (short-term, long-term, implicit, explicit), it has been difficult to identify the physical mechanisms in the brain that drive the creation, recognition, and recollection of these memories. Two types of memories crucial to high level cognition are semantic and episodic. It is widely accepted that semantic memories are stored in the neocortex, which are likely encoded in a distributed populations of neurons [29].

2.5.1 Semantic Memories

An individual's knowledge of the world is based on acquired facts: concept attributes, concept behavior, interactions between individuals, the meaning and relationships between words. Semantic memory is the memory of acquired facts, and permits

the retrieval of information without modification [30]. Semantic memories could be either simple or complex in structure, but reflect an understanding of the world. These acquired facts are not restricted to a single modal of data, and in fact require many data modalities to achieve a thorough understanding of the world.

There are two approaches for evaluating semantic memory according: there is the study of semantic memory structure and the recall of semantic memory. Experiments that study the structure of semantic memory are not concerned with accuracy of the the semantic memories, and instead these experiments rely only on the subjects output. However, experiments that study the recollection of semantic memories do evaluate the accuracy, and subjects time for recollection are recorded.

2.5.2 Episodic Memories

It is important to provide a contrast to explain what semantic memory is not; episodic memory is autobiographical memory that is unique to personal experience. An episodic memory is encoded in a temporal relationship to other episodic memories, and each episodic memory contains many semantic facts. Episodic memories are relative to the individual experience, and are autobiographical in nature. While episodic memory is thought to arise in the Hippocampus, the Neocortex plays a role by providing semantic information for the construction of an episode [31]. The Neocortex is responsible for the consolidation or binding of episodic memories into distributed circuit for long-term storage [32].

Most AI systems today do not make use of episodic memories. The relationship between semantic and episodic memory is coupled, but the relationship is mostly one way. Semantic memory can operate independently of episodic memory with a few exceptions.

2.6 Representation Similarity Analysis

A known challenge in Neuroscience is the mapping of computational models of neural circuits to physical brain-activity. Techniques for acquiring empirical brain-activity include fMRI, scalp electrophysiology EEG, MEG, all of these methods provide different granularity. Scalp electrophysiology can record the electrical activity of individual cells, whereas fMRI measures the hemodynamics of brain regions. Activity overlap is expected in both cases, but it is difficult to establish a one-to-one mapping of the different scales and data modalities. The same problem arises again when trying to find mappings between computational models and empirical data from the brain. A solution to this mapping problem is Representational Similarity Analysis, which focuses on the comparing only the similarity of data in each unique domain [33].

The characteristic metric in Representational Similarity Analysis (RSA) is the Representational Dissimilarity Matrix. The Representational Dissimilarity Matrix (RDM) is a symmetrical matrix containing a measure of dissimilarity between activity patterns. Each cell in the matrix corresponds to the level of dissimilarity (similarity when inverted?) between all activity pattern pairs for two given stimulus. The matrix is symmetrical along a diagonal of zeros in an ideal case. The measure of distance or dissimilarity function is typically the correlation-1 (Pearlson's correlation) [33][34][35].

Chapter 3

Research Methods

The first research question, ***What are the metrics for measuring the quality of semantic content for Sparse Distributed Representations?***, was addressed with a uniqueness matrix and t-SNE plots. Both these methods focus on the semantic content of Sparse Distributed Representations (SDRs) in HTM. The uniqueness matrix is an extension of Mnatzaganian's uniqueness metric for SDRs, which establishes a specific value for the similarity between any two SDRs. The second evaluation technique, t-SNE, was used for visualizations of any semantic clusters in the SDR data. A comparison between the uniqueness matrix and the t-SNE techniques was implemented throughout all experiments to show the tradeoffs between the two visualization techniques.

The second research question, ***How can spatial semantic information in images be encoded into binary representations for the Hierarchical Temporal Memory's Spatial Pooler?***, was addressed by the use of the Contractive Autoencoder as an encoder for HTM. The Contractive Autoencoder (CAE) was evaluated for its capability to extract spatial semantic information in images and its compatibility with the HTM Spatial Pooler. The CAE was used to extract image features from grayscale images on the MNIST, Fashion-MNIST, and small NORB datasets. The performance of this encoding technique was compared with the thresholding of raw image values. The best configurations of the CAE for the Spatial Pooler

were determined experimentally by using the uniqueness matrix and t-SNE plots to analyze the Spatial Pooler SDRs.

The third research question, ***How can vector operations of Sparse Distributed Representations be enhanced to produce separable representations?***, was addressed by the implementation of an SDR binding operation. The binding operation was inspired by neuroscientific theory describing interaction between the neocortex and hippocampus, and also inspired by the binding operation in Vector Symbolic Architectures. The binding operation was demonstrated on the MNIST dataset for separability. The binding operation was also demonstrated with the small NORB dataset by superimposing a set of bound SDRs together. This was done to demonstrate how more complex SDRs might be created from simpler SDRs. In addition a small exploration was done to study the density of SDRs for superimposition.

The overall system architecture with the Contractive Autoencoder, Spatial Pooler, Evaluation Metrics, and vector operations are shown in Figure 3.1.

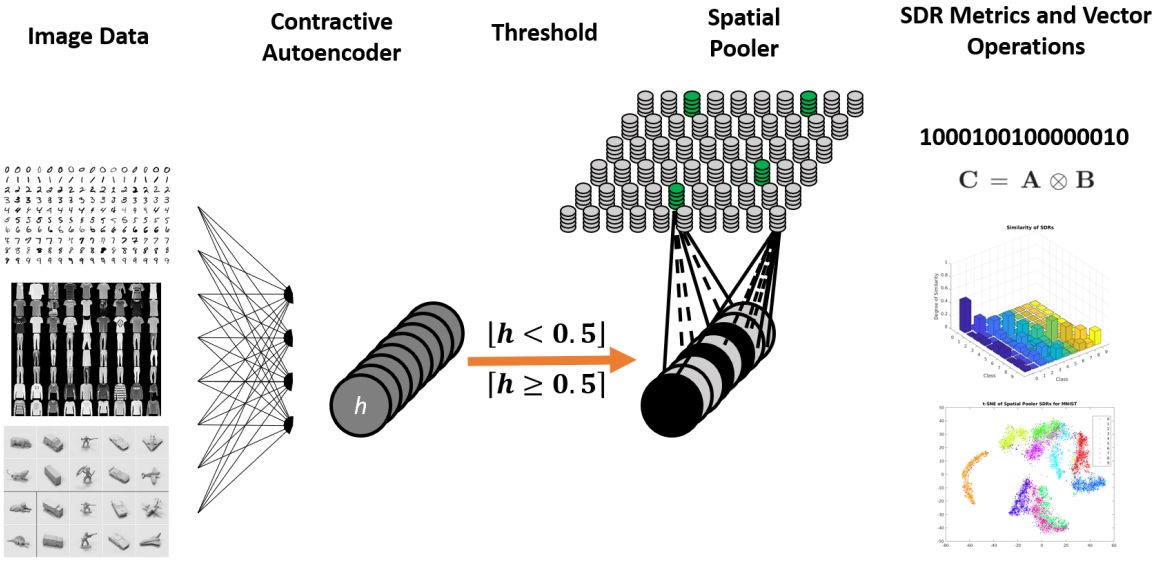


Figure 3.1: The encoder, Spatial Pooler, and evaluation metrics.

3.1 t-SNE and Uniqueness Matrix

Dimensionality Reduction techniques for visualizing high dimensional is exceptionally useful for gaining insight into raw datasets, debugging representations in machine learning algorithms, and providing intuition for preprocessing. Applying dimensionality reduction techniques to HTM systems will provide this insight into the structure of the raw dataset, the structure of the data after it is encoded into a binary representation, and the structure of the Spatial Pooler's SDRs. There are two types of dimensionality reduction techniques: those that preserve local structure and those that preserve global structure. This work uses t-SNE [27] for the visualization of the global structures of raw data, encoded data, and SDRs. Because t-SNE is an optimization process, convergence is influenced by the hyperparameters. The effect of these hyperparameters can have a drastic effect on the visualization and reproducibility [28].

3.1.1 Measurement of Similar SDRs

The computation of similarity (SDR overlap) can be computed by taking the dot product of two SDRs. Mnatzaganian's work on Spatial Pooler proposed a novel metric based on SDR similarity for evaluating the similarity of the representations [36]. The similarity value or overlap metric, mu, indicates the degree of similarity for a set of SDRs. This value is computed by taking the average overlap value and dividing it by the maximum overlap (3.3). The average overlap for a set of n SDRs is computed by (3.1), and the maximum overlap is computed by (3.2). A μ value of *1* indicates that all the SDRs are identical and a value of *0* indicates that all the SDRs are unique.

$$ao = \frac{2\sum_{s=0}^{n-2}(\sum_{u=s+1}^{n-1}(W_s \bullet W_u))}{n(n-1)} \tag{3.1}$$

$$mo = kmax\left(\sum_{i=0}^{m-1} W_{s,i} \forall s, 2\right) \tag{3.2}$$

$$\mu_o = \frac{ao}{max(mo, 1)} \tag{3.3}$$

This work utilizes the uniqueness metric to construct a uniqueness matrix for datasets with class labels. The process for constructing the uniqueness matrix is dependent on labeled data, which is used to compute $c(c-1)$ uniqueness metrics for all possible pairs of SDRs for two classes. In (3.2) W is defined as the union of SDRs from two classes. The uniqueness matrix displays information about the global structure of data by illustrating the similarity between different classes. In addition it also displays information concerning the local structure of the data. The local structure is shown by the diagonal in similarity matrix, indicating how similar SDRs within the same class appear. Because of all the possible $c(c-1)$ class pairs, the matrix is symmetrical along the diagonal axis.

3.1.2 Visualizing the Geometry of SDRs

It is important to observe the effects vector operations on SDR geometry that occur when SDRs are bound and superimposed together. If the geometry of the SDRs change due to vector operations, then the data clusters of the points may change as a result. While methods like t-SNE or UMAP are state of the art for global structure [37], the visualizations rely on a preconceived notation of neighboring points or number of classes (perplexity parameter for t-SNE and k-nearest neighbors algorithm for UMAP). However, the neighboring number of points may change as the representational structures (SDRs in this case) are manipulated by vector operations, so methods that capture the geometry without a preconceived notion for the shape of the clusters will better track the effects of the vector operations. In the t-SNE

algorithm there are inconsistencies associated with creating the embeddings. The optimization process required to find the best embedding is computationally expensive, and is not guaranteed to give the same results for subsequent runs. There are various hyperparameters that can be modified for the t-SNE algorithm, and different hyperparameters will give different clusters in the visualization. It is required to create multiple t-SNE plots with various perplexity levels to reveal the correct clusters in the data [28].

Instead of relying on dimensionality reduction techniques, this work explores using Representational Similarity Analysis to visualize the geometric space. There is no optimization process or hyperparameters in Representational Similarity Analysis. Instead only a distance function is used to compare data points. This will allow for more consistent representations than t-SNE. The construction of Representational Similarity/Dissimilarity Matrices will be beneficial for evaluating SDR content by observing the relationships among SDRs. There are two major benefits with this approach to representation evaluation: RSA provides a verification technique for HTM and also provides a way to observe high-dimensional geometric changes in the data. Because HTM is a biologically inspired algorithm, it can be compared to neocortical regions it is modeled after. In this manner the Sparse Distributed Representations in the HTM model can be compared to empirical neocortical data, when both the model and the brain subject are given the same stimulus. In this manner the matrices constructed from the Sparse Distributed Representations and empirical brain data may be compared to ensure that the same degree of relationships exist in both domains. This work only *suggests* RSA as verification technique for HTM models, and instead focuses on utilizing the matrices described in RSA to observe how representational relationships and the geometry of high dimensional vectors (SDRs) change. The RSM was used to model the effectiveness of the binding operation.

RSA has been utilized to understand the physical representations in the brain for

semantic memories [34][38][29].

3.2 CAE as an encoder for the HTM Spatial Pooler

The second contribution of this work is the investigation of an encoder for the Spatial Pooler that preserves the semantic information and spatial relationships of various grayscale image datasets. This work proposes the Contractive Autoencoder (CAE) as an image encoder for HTM because it has been shown that the Jacobian regularization term encourages saturated (nearly binary) representations [25][23]. The loss function of the CAE penalizes incorrect reconstructions of the input samples, so the hidden representations must contain semantic information concerning differences between different input samples. The design of the CAE is focused on obtaining robust representations instead of robust reconstruction [39], so the CAE does not function as a lossless encoder. Several methods were used to analyze the performance of the CAE. These metrics consisted of a histogram of activation values after training, plots of optimization loss during training, comparisons between the reconstructed images and raw images, and t-SNE plots of the hidden layer representations after training.

Although the CAE produces nearly binary representations, the Spatial Pooler in HTM requires strictly binary inputs. The saturated representations were then rounded to binary values in order to be compatible with the Spatial Pooler as shown in Figure 3.1. It is expected that thresholding of the activation values will alter the representation content in some manner, so this compares the saturated representations with the binary representations. This was primarily done by using the decoder in the CAE, shown in Figure 2.10, to reconstruct the input and use t-SNE algorithm to explore the structure of the representations.

While the CAE was not designed with the intention of producing saturated or binary values, this work explores which configurations of the CAE are ideal compatibility as a Spatial Pooler encoder. The CAE and Spatial Pooler system, shown in

Figure 3.1, was evaluated based on three grayscale image datasets. The best CAE configuration was determined by varying the regularization values of the CAE for each datasets. After the CAE models were trained, all activation values for the training and testing sets were made binary by rounding. The binary representations created by the CAE model were then used as the input to HTM Spatial Pooler. Several instances of the Spatial Pooler were created and trained on the hidden layer activations to observe the effect of different HTM hyperparameters. This was repeated for each dataset. After the training of the Spatial Pooler, a t-SNE plot and uniqueness matrix were generated for the Spatial Pooler SDRs to visualize the semantic relationships. The Spatial Pooler was also evaluated in the same manner on the three datasets with a simple thresholding encoder method. For this system, the grayscale images were thresholded to binary values.

3.2.1 Image Datasets

The datasets used for the CAE and Spatial Pooler were MNIST, Fashion MNIST, and a subset of the small NORB dataset. The MNIST dataset consists of grayscale images of handwritten digits [40]. The labels for each image directly correspond to the digits. It has been for classification of images in [8][1]. The Fashion MNIST dataset was created in response to high classification accuracies reported on the MNIST datasets [41]. Because of the high classification accuracies, and is considered a solved problem. The MNIST dataset is overuseed, and it is not representative of modern computer vision tasks. The Fashion-MNIST addresses theses issues by providing a dataset that is focused on fashion products. The categories for Fashion-MNIST are shown in Table 3.1. Both the MNIST and Fashion-MNIST datasets consist of 60,000 training images and 10,000 test images.

The third dataset was a subset of the small NORB dataset. The small NORB

Table 3.1: Fashion-MNIST Labels

Labels	Description
0	T-Shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle Boot

dataset contains stereoscopic grayscale images of toy objects with a consistent background, where each object was imaged under six lighting conditions, nine elevations, and eighteen azimuths [42]. There are five categories and five instances of each object for both the train and testing split. The labels for each of the categories is shown in Table 3.2.

Table 3.2: Small NORB Labels

Labels	Description
1	Four-Legged animals
2	Human Figures
3	Airplanes
4	Trucks
5	Cars

The entire NORB dataset contains 24,300 images for both the train and test split. The subset that was utilized in this work consisted only of two elevations, one lighting condition, and all eighteen azimuths. All five categories and instances were used, so the total number of images in the dataset were 900 images. The only two elevations used were a side on perspective and a top down perspective. The subset was chosen because there is a considerable amount of variation in the small NORB images.

3.3 Binding of Sparse Distributed Representations

The SDR vector operations can be enhanced by looking at the brain for a source of inspiration. There are theories that suggest the binding of neuron activations occur in the brain for memory consolidation. The neocortex and hippocampus work closely together to create episodic memories, which is accomplished by binding together cortical neurons into a coherent representation or memory trace [32][43][31][44]. Theories of the hippocampus function suggest that it binds together the multimodal and multidomain representations that are created in the neocortex, and also binds objects with their spatiotemporal contexts together [31]. According to this theory, binding is critical for the integration of multiple concepts and multimodal representations to create mental representations for language and vision [16][13]. These theories focus on neocortex and hippocampus interaction, but the focus of HTM is on the neocortex alone. It is important to understand how the neocortex functions with other brain regions, as the neocortex is not the only region in the brain. In fact the role of the hippocampus is that it processes novel information that the neocortex cannot, which was stated in *On Intelligence* (pg. 168-171) [4]. The hippocampus can quickly store patterns and has the ability to recall these novel patterns. The binding of HTM's SDRs would be a direction towards this type of rapid memory storage and processing that is found in the hippocampus.

Binding is a significant vector operation the implementation of a Vector Symbolic Architectures (VSAs). Binding in VSAs is crucial for the symbolic processing, which allows the reference, assignment, and query of variables. Variable binding is essential for distributed representations in the brain because it provides a way to generalize data that is not possible with long or short term potentiation [45]. The high dimensionality, fixed length, distributed nature, sparseness of HTM's SDRs strike similar comparisons to the existing VSAs (HHR, BSC, MAP). The analysis of HTM's SDRs

was based on the intuition of the binary representations proposed by Kanerva's work on Sparse Distributed Memory and Hyperdimensional Computing [10].

The combination of VSA theory and HTM's Temporal Memory algorithm was explored in [46] to determine if Temporal Memory was able to bind information. The approach to binding was the presentation of binary vector pairs in sequence to the HTM region during training, so that either vector of the pair would predict the other. The ordering of the pairs was swapped throughout learning to achieve the binding between both binary vectors. The results showed that unbinding was highly accurate with a large number of columns. In this approach binding was demonstrated that binding can be learned implicitly in the distal synapses. This in line with the theory that the binding of representations can be a created through Hebbian learning [16][47].

This implicit binding between binary vectors is driven by changes in synaptic weights, which may pose challenges for HTM as a continuous online learning system. It is unclear from [46] how many times the pairs were presented to the region to achieve the stated performance on binding and unbinding. In order for the binding operation to be learned implicitly in the distal synapses, the pairs of random binary vectors would have had to be presented several times to the region as the distal synapses are initialized below the connection threshold. Instead this work explores an explicit binding operation for the SDRs in HTM.

3.3.1 Explicit Binding Operation and Spatial Pooler

The method outlined here for SDR binding is based on the work of Laiho et al's generalized implementation of the HRR's circular convolution in the frequency domain for sparse binary vectors [48]. This method divides the sparse binary vector into contiguous segments of bits. There are S of these segments in a sparse binary vector with N bits, and thus each segment contains $L = N/S$ bits. If there exists only

one bit in each segment, then the vector is considered *maximally sparse* as shown in Figure 3.2.

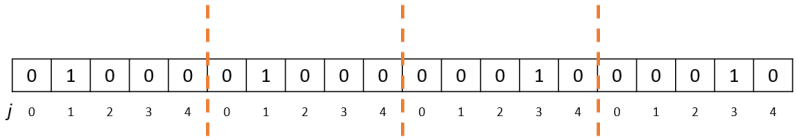


Figure 3.2: A 20 bit ($N = 20$) maximally sparse vector. The vector is divided into four segments ($S = 4$) of five bits each ($L = 4$). The vector is maximally sparse due to the presence of a single *on* bit in each segment.

The binding operation between two vectors is implemented with a segment-wise bit shift; the index of the first nonzero bit in the first vector's segment determines how many bits to shift the bits in the first segment of the second vector. This process is repeated for each of the remaining segments in the vector, an example of the binding operation is shown in Figure 3.3. However, if the bits in the first vector are not maximally sparse, then additional operations are required to produce a maximally sparse vector. There are two solutions for this scenario: pick one of the bits at random or take the bit index of the modulo sum of all bit indices in the segment.

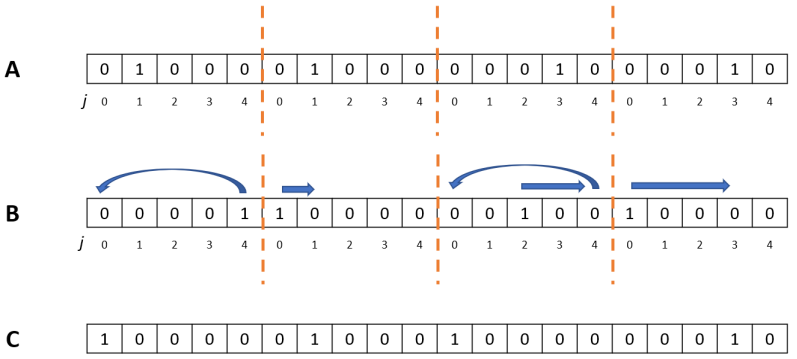


Figure 3.3: The binding (circular convolution) of SDR A & B produce an SDR C ($C = A \otimes B$). All SDRs have 20 total bits ($N = 20$), four segments ($S = 4$), and five bits per segment ($L = 5$).

3.3.2 Creating Robust and Separable Representations for NORB

Because a binding operation is useful temporal data and complex memory structures, HTM could benefit from a binding mechanism for SDRs. Numenta's recent work on object understanding in HTM explains that location signals are combined with a sensory signals, which is then presented to the HTM model [7]. Within this framework a memory structure could be created to preserve both the sensory information and location information concerning a spatial object with one SDR.

The binding operation was used to create more compact and separable representations for spatio-temporal data. This was demonstrated on subset of the small NORB dataset. While this dataset is intended to be used for object classification, this work utilizes small NORB to illustrate how features can be combined with azimuth and elevation information to create compact representations. Numenta has stressed the importance of location signals in the HTM structure for robust object representations, which are used with Temporal Memory make predictions about object features as it is manipulated [7].

The CAE was used to encode NORB images at two elevations and all 18 azimuths to create binary representations for the Spatial Pooler. The binary representations of the small NORB images were used to train the Spatial Pooler in a spatiotemporal manner. The effect of rotating a NORB object, at either elevation, in space counter clockwise was done presenting the binary representations to the spatial pooler in order of increasing azimuth values. The location signals/SDRs were created by generating 18 random maximally sparse SDRs with the same sparsity properties as those produce by the Spatial Pooler, which were used to represent the 18 different azimuth values. After the Spatial Pooler was trained on the binary representations of the images, all binary representations for each object were passed through the Spatial Pooler in the same manner to create SDRs for the features at all azimuths. These azimuths were then bound together with the corresponding location signal to create 18 bound SDRs.

The set of bound SDRs, b, are created in (3.4), where the $\otimes$ is binding operation shown in Figure 3.3. All bound pairs were then superimposed to create an unordered set, r, containing all the information regarding the spatiotemporal information (3.5). The sum operation in (3.5) is a logical OR operation.

$$b_i = c_i \otimes l_i \tag{3.4}$$

$$R = \sum_{i=0}^{18} b_i \tag{3.5}$$

The t-SNE visualizations and uniqueness plots were used to understand the effects of the vector operations, and how combining data in this manner would be beneficial for the representations.

Chapter 4

Results & Analysis

The results for the proposed encoding method and the proposed implementation of the Binding operation are reported in this chapter. The baseline performance of the Spatial Pooler is reported in Section 4.1.1, the results of the CAE and the Spatial Pooler are reported in Section 4.1.3, and the desired configuration and performance of the CAE is reported in Section 4.1.2. The hyperparameters used for the models are listed in Section 4.1.5. The observations are discussed in more detail in Section 4.2.

4.1 Results

4.1.1 Spatial Pooler

This section includes the results for the Spatial Pooler when using a thresholding technique for encoding image data. A different Spatial Pooler was trained on each dataset. The Spatial Pooler was trained and evaluated on the MNIST, Fashion MNIST, and a special subset of the NORB dataset. All Spatial Poolers were trained for one epoch over the complete training set. All plots were generated from the testing set, except for the NORB subset where the plots were generated from the training set. The significant plots used were a uniqueness matrix and t-SNE plot of SDRs for the purpose of evaluating the semantic similarity and relationships. The grayscale pixel values of the NORB subset were normalized to values between 0 and 1. The MNIST and

Fashion MNIST datasets were already normalized to values between 0 and 1. The thresholding technique consisted of rounding the image values to 0 and 1 (i.e. values less than 0.5 were set to 0, and values greater than or equal to 0.5 were set to 1).

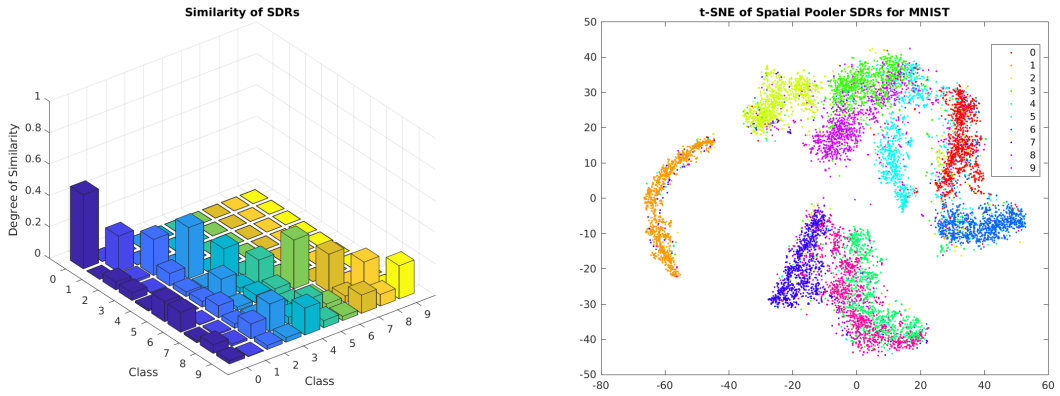
4.1.1.1 MNIST

The uniqueness plot of the SDRs is shown in Figure 4.1a; the diagonal of this matrix indicates the similarity between SDRs of the same class.

The t-SNE plot of the Spatial Pooler SDRs are shown in Figure 4.1b. This shows how some digits are more similar in their representations than other digits. The digits *3,5,8* form a cluster in the top center of the plot (green, cyan,purple); the digits *4,7,9* form a cluster in the bottom center (forest green, dark blue, magenta). It appears that the Spatial Pooler is creating similar SDRs for semantically similar classes. This is also illustrated by the uniqueness plot in Figure 4.1a. The bottom horizontal row in the uniqueness matrix corresponds to the semantic similarity between class *9* and all other classes. The classes or sets of SDRs that are most similar to the class *9* besides itself according to the uniqueness plot are *4* and *7*. This is important because both these digits are drawn in a similar manner as the digit *9* because they all share the same general downward stroke.

4.1.1.2 Fashion MNIST

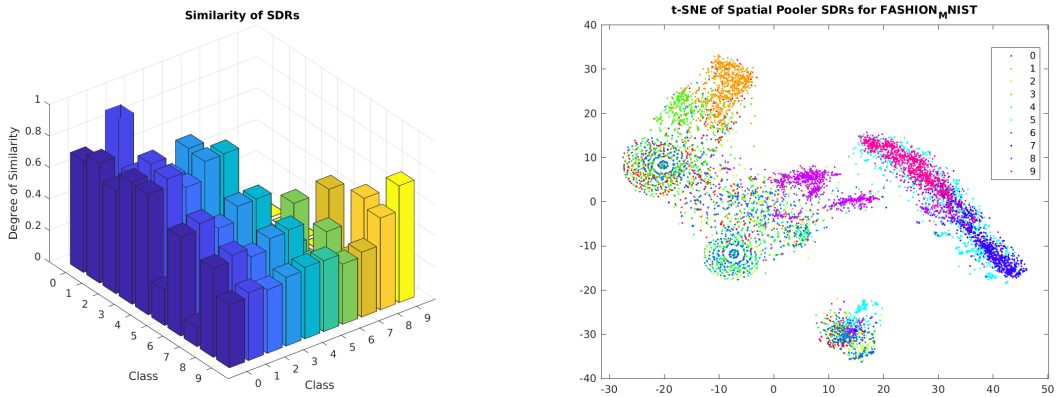
The uniqueness plot of the Spatial Pooler SDRs are shown in Figure 4.2a, which shows a much higher degree of similarity between all classes of SDRs. The last row (class 9; ankle boots) is most similar to class *8* (Bags), but there is also similarity to all 8 remaining classes. The t-SNE plot of the Spatial Pooler SDRs is shown in Figure 4.2b. The t-SNE algorithm doesn't converge to an optimal solution, and fails to effectively cluster the data points at two locations. The algorithm fails to cluster because the data points between different classes are too similar, which can be observed by the



(a) Uniqueness plot of SDRs. **(b)** t-SNE plot of SDRs.

Figure 4.1: Semantic similarity between the Spatial Pooler's SDRs for MNIST.

uniqueness plot in Figure 4.2a.



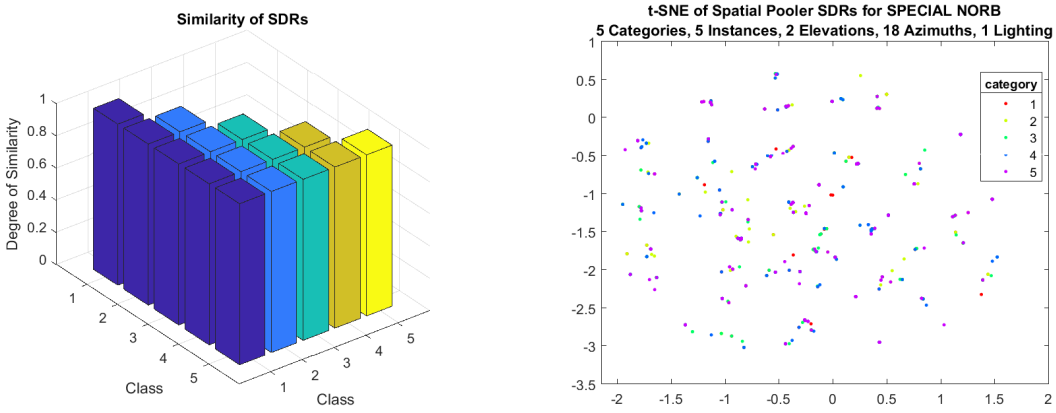
(a) Uniqueness plot of SDRs. **(b)** t-SNE plot of SDRs.

Figure 4.2: Semantic similarity between the Spatial Pooler's SDRs for Fashion MNIST.

4.1.1.3 Special NORB

The uniqueness plot is shown in Figure 4.3a, which illustrates that there is complete similarity between all five classes in the NORB dataset. The magnitude of each μ or column in the uniqueness plot is 1, meaning that the Spatial Pooler learned the same SDR for each data sample. The same information can be found when looking at the

t-SNE plot of the SDRs in Figure 4.3b. The t-SNE plot does not produce any visible clusters even though the SDRs are spaced out, which can be observed by the small range of the axis in the t-SNE plot.



(a) Uniqueness plot of SDRs. **(b)** t-SNE plot of SDRs.

Figure 4.3: Semantic similarity between the Spatial Pooler's SDRs for a subset of the small NORB dataset.

4.1.2 Contractive Autoencoder

The preprocessing techniques applied to all the images consisted of a subtraction of the mean and a division by the standard deviation among the training samples for each respective dataset. The NORB subset was resized from 96x96 pixels to 64x64 pixels. Three single layer CAEs were trained on each of the three training datasets. The MNIST and Fashion-MNIST CAEs were trained for 300 epochs, and the NORB subset CAE was trained for 150 epochs. The learning rate, α, for all CAEs was 5.0×10^{-4}. The dimension of the hidden layer was equal to the dimensionality of the input data for all datasets; the CAE for MNIST and Fashion MNIST were 784 hidden units, and the dimension for the NORB subset was 4096. Stochastic gradient descent was used for the optimization of the CAE cost function. A batch size of 128 was used for MNIST and Fashion-MNIST, and a batch size of 10 was used for the

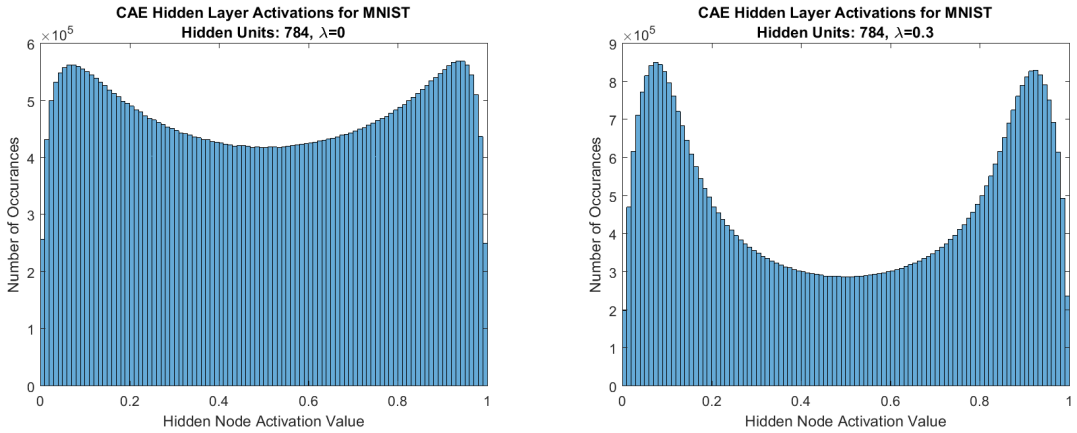
NORB subset. The regularization parameter, λ, was the only hyperparameter that was varied throughout the experiment.

4.1.2.1 MNIST

The reconstructions for MNIST are shown in Figure 4.4. The quality of the reconstructions is better with less regularization or a lower value of λ; the difference in reconstructions is shown prominently by the digits *5* and *6*. However, less regularization results in less saturated values, as shown by the plots in Figure 4.5a and Figure 4.5b.

(a) $\lambda = 0$ **(b)** $\lambda = 0.3$

Figure 4.4: Comparison of MNIST CAE reconstructions (bottom) with input images (top)



(a) $\lambda = 0$ **(b)** $\lambda = 0.3$

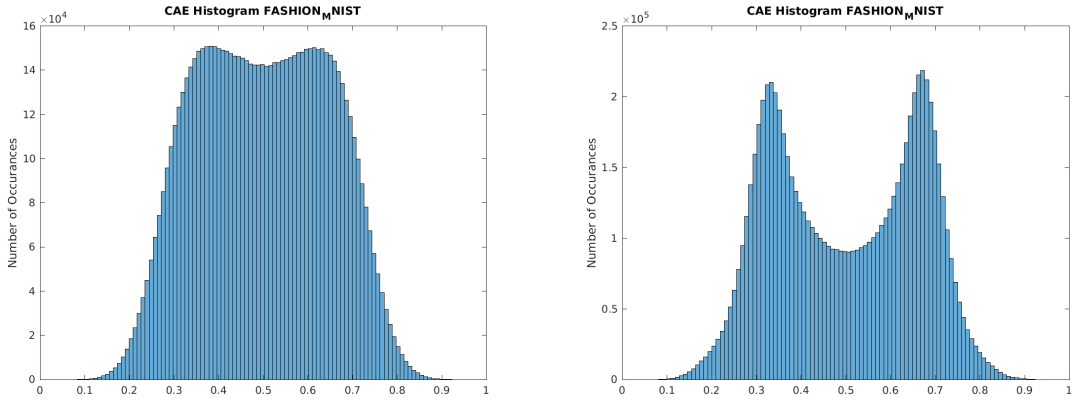
Figure 4.5: Distribution of hidden layer activation values on MNIST dataset. There are more saturated values with more regularization strength.

4.1.2.2 Fashion MNIST

The reconstructions for Fashion MNIST are shown in Figure 4.6. The reconstruction again is stronger with less regularization. The effect of strong regularization is shown by looking at the sandal class reconstruction (sixth image from the left) in Figure 4.6b, which shows the reconstructed sandal appears more like a sneaker when compared to the more accurate reconstruction in Figure 4.6a. The same phenomena is also observed by looking at the bag class reconstructions (ninth image from the left) in Figure 4.6b, which again shows that the reconstructed bag appears more like a ankle boot when compared to the more accurate reconstruction in Figure 4.6a.

(a) $\lambda = 0$ **(b)** $\lambda = 0.6$

Figure 4.6: Comparison of Fashion MNIST CAE reconstructions (bottom) with input images (top)



(a) There is not a clear divide between activation values. $\lambda = 0.01$ **(b)** There is a clear divide between the activation values. $\lambda = 0.6$

Figure 4.7: Distribution of hidden layer activation values on FASHION MNIST test dataset. There are more saturated values with more regularization strength.

4.1.2.3 Converting Saturated values to Binary

It is important to observe what information is lost by thresholding the saturated value to binary. This was done by comparing the reconstructed images for saturated to the binary representations for the CAE. The reconstructed images in Figure 4.8 of the digit 5 that are semantically similar to the digit 3. These results were found by using a classification technique (softmax and SVM) to find instances were the classifier predicted the digit 3 instead of 5. The saturated and binary representations were compared for the same input samples. When examining the input and reconstructed images of the digit 5 on the far left in both Figure 4.8a and Figure 4.8b, the binary reconstructed representation appears more like the digit 3 than the digit 5.

The same reconstruction technique was used for the Fashion MNIST datasets, and specifically samples of ankle boots (class 9) that were similar to bags (class 8). The reconstructions are shown in Figure 4.9, but is difficult to observe or identify the spatial features that correspond to a bag even with the saturated values.

(a) Saturated Representations **(b)** Binary Representations.

Figure 4.8: Comparison of MNIST images (top) with CAE reconstructions (bottom).

(a) Saturated Representations. **(b)** Binary Representations.

Figure 4.9: Comparison of Fashion MNIST images (top) with CAE reconstructions (bottom).

4.1.2.4 Regularization Values

The regularization values that were chosen for CAEs for the MNIST, Fashion MNIST, and NORB subset datasets are $\lambda = 0.3$, $\lambda = 0.01$, and $\lambda = 0.01$ respectively. The MNIST and Fashion MNIST regularization values were chosen to balance reconstruction error while maintaining a significant amount of saturation. The lambda value for the NORB subset was significantly weaker due to the total number of weights in the network. A weaker lambda value ensures that the representations contained enough information to discern between different sample categories.

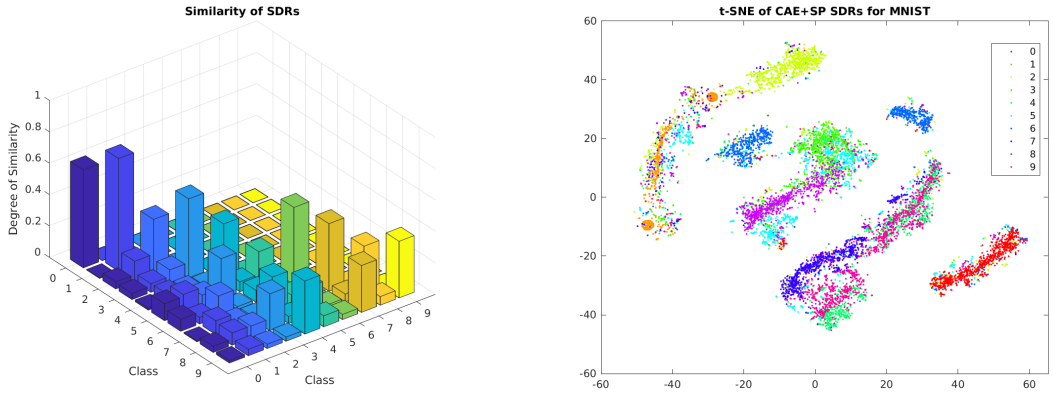
4.1.3 CAE & Spatial Pooler

The compatibility of the CAE's activations were evaluated with the Spatial Pooler; the results in this section show how the Spatial Pooler performs with the CAE as an encoder.

4.1.3.1 MNIST

The uniqueness plot of the test representations are shown in Figure 4.10a. The similarity of SDRs from the same class is much greater than the basic image thresholding technique, which can be seen by comparing the diagonals of Figure 4.1a and Figure 4.10a. The t-SNE plot of the SDRs is shown in Figure 4.10b, which now shows more entangled relationships. The SDRs corresponding to the digits *3,5,8* (light green, cyan, purple) are clustered in the center, and just below that is a hook shaped cluster for the digits *4,7,9* (green,dark blue, magenta). The orange cluster on the far left corresponds to SDRs of the digit *1*. There are two orange circles at the top and bottom of the orange cluster that resulted due to the t-SNE algorithm failing to cluster effectively. The similarity between the SDRs representing the digit *1* are too similar, so the t-SNE algorithm cannot create enough neighbors for the points. This is also supported by the uniqueness plot by looking at the intersection of the *1*

class with itself in the top left corner of Figure 4.10a: the μ value is the largest in magnitude for the entire matrix.



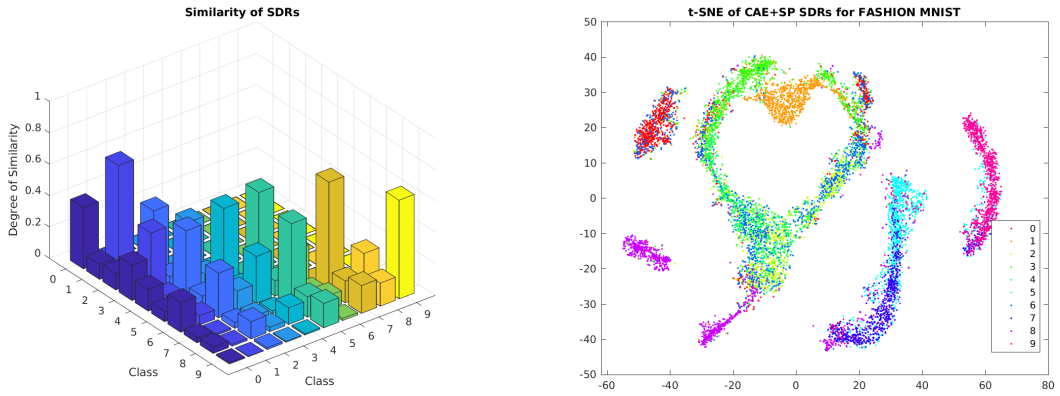
(a) Uniqueness plot of SDRs. **(b)** t-SNE plot of SDRs.

Figure 4.10: Semantic similarity between the Spatial Pooler's SDRs for MNIST.

4.1.3.2 FASHION MNIST

The uniqueness plot of the test representations are shown in Figure 4.11a; the similarity between the classes with CAE shows clearer relationships. The last horizontal row represents the semantic relationships for the ankle boot class in the uniqueness matrix, and it has the strongest similarity or largest μ to itself (class *9*). The ankle boot SDRs also have lesser semantic relationships to sandals and sneakers (class *5* and *7*), but also has a small semantic relationship to bags (class *8*).

The t-SNE plot of the Fashion MNIST SDRs are shown in Figure 4.11b. The cyan and dark blue cluster to the left of the far right magenta cluster is primarily composed of two classes: the sneaker SDRs on bottom (dark blue, class *7*) and the sandal SDRs on the top (cyan, class *5*). The reconstruction image of the sandal in Figure 4.6b shows that appears more like a shoe than the input sandal. This is likely why the two classes are clustered so close to each other. However, there are some outliers in the bottom left of the cluster that consist of bags, ankle boots, and sneakers.



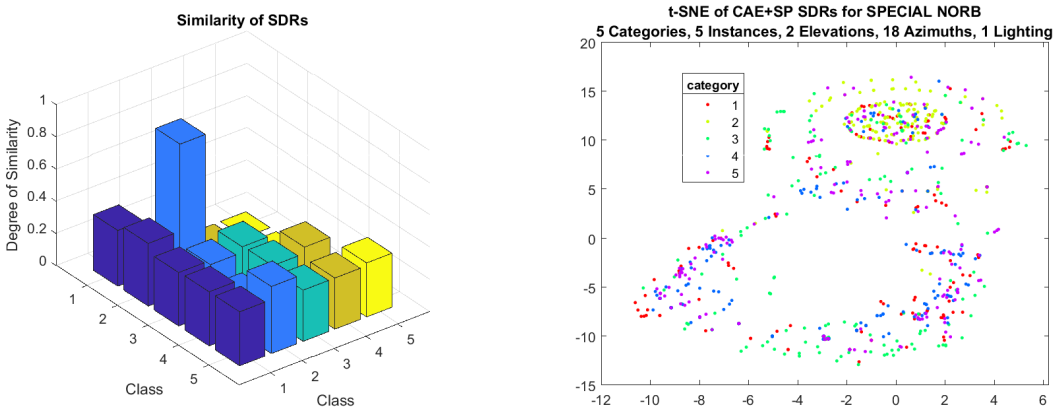
(a) Uniqueness plot of SDRs. **(b)** t-SNE plot of SDRs.

Figure 4.11: Semantic similarity between the Spatial Pooler's SDRs for Fashion MNIST.

4.1.3.3 Special NORB

The uniqueness plot of the special NORB in Figure 4.12a has a similar level of dissimilarity between SDRs of all classes except for class *2* (human toy figures). The large uniqueness value for class *2* indicates that the representations for this class are similar, and is also illustrated by looking at t-SNE plot of the same data in Figure 4.12b. The concentration of class 2 (yellow-green) is located in the top "ring" of the t-SNE plot, and this implies that the class *2* SDRs are semantically similar for both elevations of the objects. The large amount of variance in the positions and angles found in the small NORB datasets are likely responsible for the lack of any significant clusters in the t-SNE plot.

It is important to observe that labeling has a significant effect on both the uniqueness plot and t-SNE plots. The labels used for both plots are only the category labels. In turn more detailed combinations of labels could have been used between for both plots such as label illustrating the category and elevation.



(a) Uniqueness plot of SDRs. **(b)** t-SNE plot of SDRs.

Figure 4.12: Semantic similarity between the Spatial Pooler's SDRs for a subset of the small NORB dataset.

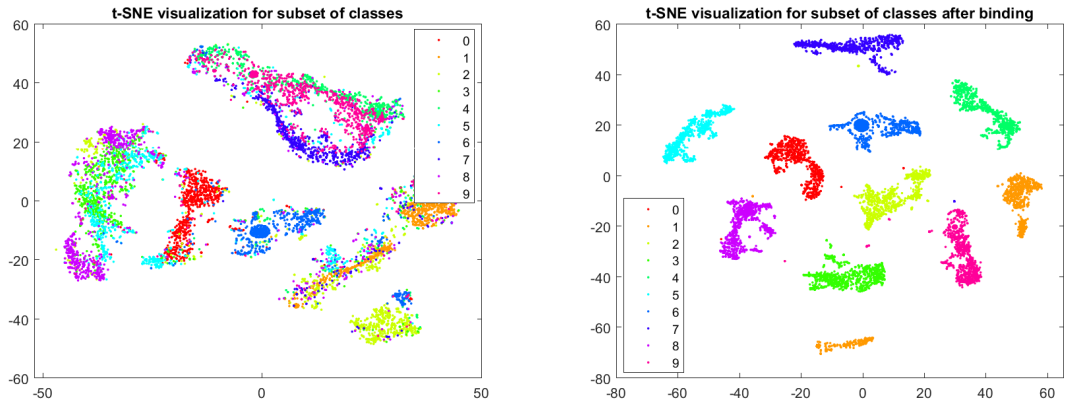
4.1.4 SDR Binding and Superposition

The previous results of this section show that stronger semantic relationships are created between SDRs with the CAE. These SDRs were used to observe the effects of vector operations. The SDRs were manipulated by the superposition and proposed binding operation. The goal of these experiments is to examine how these operations might untangle the semantic relationships to produce separable representations.

4.1.4.1 Binding with MNIST

The result of the binding operation between 10 random maximally sparse SDRs with each class of SDRs is shown in Figure 4.13b, which shows that this operation untangles and removes the semantic relationships that existed before in Figure 4.13a. Each SDR is clustered along with the other SDRs for a particular class, but there are some outliers where the SDRs after binding are closer to other class clusters. It should be noted that it is possible to go back to the original space in Figure 4.13a by performing a binding operation between the bound SDRs and the 10 maximally sparse cue SDRs. This binding experiment was configured such that the distances between SDRs from

the same class is maintained through the binding operation, which is accomplished by binding the SDRs of each class with randomly generated maximally sparse SDR. This is not observable through the t-SNE plots in Figure 4.13, so a representational similarity matrix was used to observe the change in distances before and after binding.

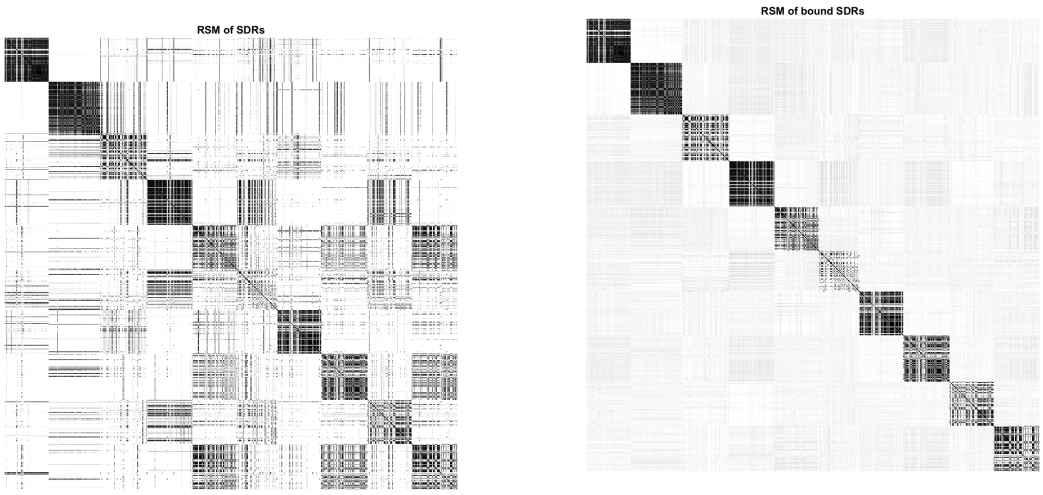


(a) MNIST SDRs **(b)** SDRs after Binding

Figure 4.13: t-SNE plots of MNIST SDRs before and after binding.

The matrices in Figure 4.14 show the difference in distance (dot product) between all 10,000 MNIST SDRs before and after binding. The values in the matrix indicates how similar two particular SDRs are to each other: the similarity corresponds to the darkness of point where black pixels indicate high similarity and white pixels indicate low similarity. In Figure 4.13b 10 squares emerge after binding centered along the diagonal, which correspond to the 10 distinct clusters for each of the 10 classes in Figure 4.13b. Each of the emergent 10 squares has the same relative distance to other SDRs within the same class, this was verified by taking the difference between the matrics in Figure 4.13.

It should also be noted and observed that the t-SNE plot of the MNIST SDRs in Figure 4.13a and Figure 4.10b illustrate the same data. The same SDR samples and hyperparameters were used with the t-SNE algorithm to cluster the data, yet the result of the algorithm is inconsistent.



(a) RSM of MNIST SDRs

(b) RSM of MNIST SDRs after Binding with unique SDR

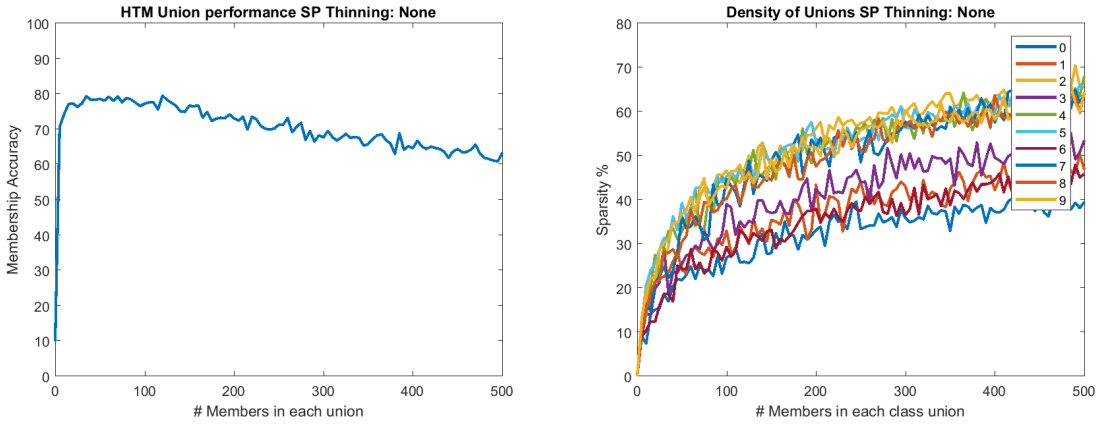
Figure 4.14: t-SNE plots of MNIST before and after binding.

4.1.4.2 Superposition with MNIST

The results for determining the membership for a superimposed vector are shown in Figure 4.15. The membership accuracy is shown in Figure 4.15a with a general decline in membership as the size of union grows. As the number of SDRs that are superimposed increases, the accuracy for determining if a particular SDR belongs to the set decreases. This is explained by looking at the density or sparseness of the union SDRs in Figure 4.15b. There is a sharp increase in the density, and the union SDRs are no longer around 2% sparsity. As the the number of SDRs that are superimposed increases, the number of active bits in the superimposed vector increases dramatically.

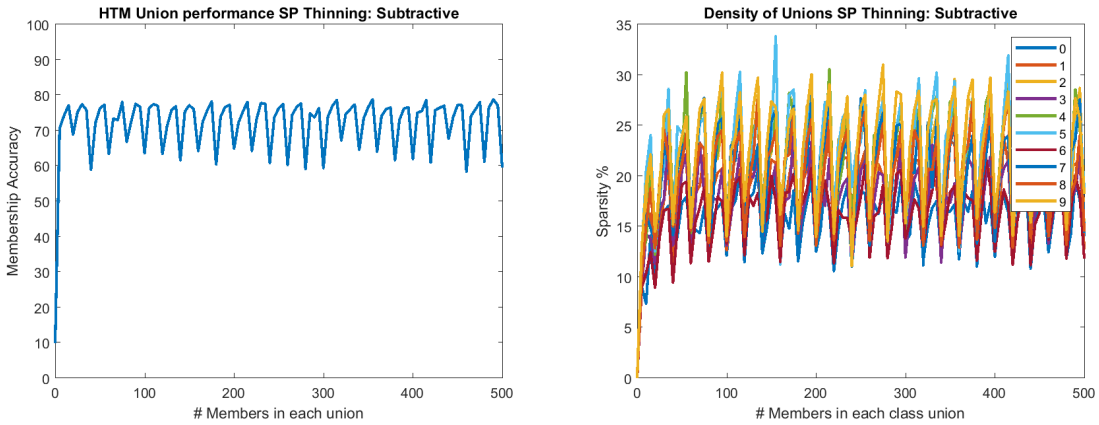
The implementation of the subtractive thinning technique reduces the decline in accuracy and the substantial increase in density. The subtractive thinning of the same superimposed vectors maintains a consistent density regardless of how many vectors are superimposed together, which is shown in Figure 4.16b. The density does not approach the target 3% sparsity, and instead the density ranges from 12% to 34%.

Figure 4.16a shows that the membership accuracy does not decline, but instead has consistent membership accuracy with a large variance.



(a) SDR Union Membership Accuracy **(b)** SDR Superposition Density

Figure 4.15: Membership Test of Superposition.

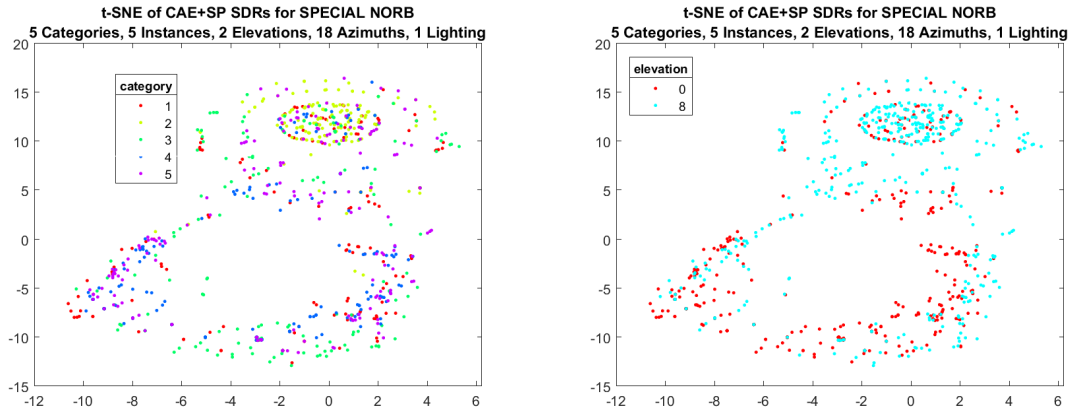


(a) SDR Union Membership Accuracy **(b)** SDR Superposition Density

Figure 4.16: Membership Test of Superposition with subtractive thinning.

4.1.4.3 Binding and Superposition with NORB

The Binding implementation was also tested in conjunction with the superposition operation for the small NORB subset. The t-SNE plots in Figure 4.17 show the SDRs for all samples in the NORB subset, which contain semantic information of images at all 18 azimuths (or angles) around each object at two elevations. Figure 4.17b only has the elevations of each point labeled, and generally the SDRs containing spatial semantic information at lower elevation (0) are clustered towards the bottom of the lower "ring" whereas the SDRs containing spatial semantic information at the top down elevation (8) are clustered in the top "ring". There are many exceptions to this observation, but examining the complex representations that result after the binding and superposition reveal more insight into these emergent clusters.



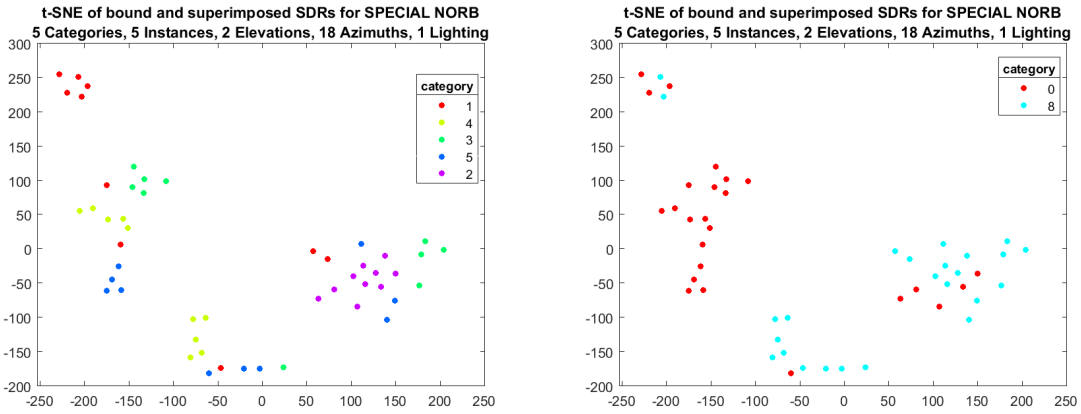
(a) t-SNE with categories labeled. **(b)** t-SNE with elevations labeled.

Figure 4.17: t-SNE plots of NORB subset at all 18 azimuths and 2 elevations.

It is important to observe that the same data exists in both Figure 4.17 and Figure 4.18, the data in Figure 4.18 is composed of bound and superimposed SDRs from Figure 4.17. The dimensionality of the SDRs is the same in both plots. The human figures (category 2, purple) in Figure 4.18a are clustered together. The same cluster is observed with the elevations labeled in Figure 4.18b, which indicates that human figures have semantically similar representations at both elevations. However,

looking at other categories such as trucks (category 4, yellow-green) in Figure 4.18a shows two main clusters of five truck SDRs, and by examining the same two clusters in Figure 4.18b the two truck clusters correspond to two different elevations.

A more general view of the complex representations (SDRs) in Figure 4.18b illustrates that the semantics in the representation at the lower elevation (0) differ from the semantics at the higher elevation (8). Generally the representations for the lower elevation exist in the upper right of the plot where as the representations for the higher elevation exist in the lower right of the plot. It should be recalled that no supervised information related to the elevation was bound to the SDRs, only the information regarding the azimuth was bound and superimposed to the SDRs in Figure 4.17b. The average density of all the data points in Figure 4.18a and Figure 4.18b is 785 active bits or around 40% density. This is much larger than the 64 active bits or 3% sparsity that the Spatial Pooler was designed to generate. This increase in density occurs because of the superposition operation, which accumulates active bits in the resulting vector.



(a) t-SNE with categories labeled. **(b)** t-SNE with elevations labeled.

Figure 4.18: t-SNE plots of bound NORB class instances at all 18 azimuths and 2 elevations.

4.1.5 Hyperparameters

This section contains the hyperparameter values for the Contractive Autoencoders and the Spatial Poolers. All Spatial Pooler's used global inhibition.

4.1.5.1 Spatial Pooler

The hyperparamters listed in this section relate to the three Spatial Poolers used for each of the three datasets with the simple image thresholding encoder.

Table 4.1: Spatial Pooler Parameters for MNIST

Parameter	**Symbol**	**Value**
Total columns in region	m	2048
Number of synapses per column	q	400
Permanence increment	ϕ_+	0.05
Permanence decrement	ϕ_-	0.05
Synapses connection threshold	ρ_s	0.5
Column activation threshold	ρ_d	5
Active bits in SDR	ρ_c	64

Table 4.2: Spatial Pooler Parameters for Fashion-MNIST

Parameter	**Symbol**	**Value**
Total columns in region	m	2048
Number of synapses per column	q	100
Permanence increment	ϕ_+	0.05
Permanence decrement	ϕ_-	0.05
Synapses connection threshold	ρ_s	0.5
Column activation threshold	ρ_d	5
Active bits in SDR	ρ_c	64

Table 4.3: Spatial Pooler Parameters for small-NORB

Parameter	Symbol	Value
Total columns in region	m	2048
Number of synapses per column	q	100
Permanence increment	ϕ_+	0.005
Permanence decrement	ϕ_-	0.005
Synapses connection threshold	ρ_s	0.5
Column activation threshold	ρ_d	5
Active bits in SDR	ρ_c	64

4.1.5.2 Spatial Pooler with CAE

The parameters listed in this section are for the combined CAE and Spatial Pooler systems for each of the three datasets.

Table 4.4: CAE Parameters for MNIST

Parameter	Value
Number Hidden Units	784
Learning rate (α)	5e-4
Regularization (λ)	0.3
Batch Size	128
Epochs	300

Table 4.5: Spatial Pooler Parameters for CAE MNIST

Parameter	Symbol	Value
Total columns in region	m	2048
Number of synapses per column	q	200
Permanence increment	ϕ_+	0.05
Permanence decrement	ϕ_-	0.05
Synapses connection threshold	ρ_s	0.5
Column activation threshold	ρ_d	5
Active bits in SDR	ρ_c	64

Table 4.6: CAE Parameters for Fashion-MNIST

Parameter	**Value**
Number Hidden Units	784
Learning rate (α)	5e-4
Regularization (λ)	0.01
Batch Size	128
Epochs	300

Table 4.7: Spatial Pooler Parameters for CAE Fashion-MNIST

Parameter	**Symbol**	**Value**
Total columns in region	m	2048
Number of synapses per column	q	100
Permanence increment	ϕ_+	0.05
Permanence decrement	ϕ_-	0.05
Synapses connection threshold	ρ_s	0.5
Column activation threshold	ρ_d	5
Active bits in SDR	ρ_c	64

Table 4.8: CAE Parameters for small-NORB

Parameter	**Value**
Number Hidden Units	4096
Learning rate (α)	5e-4
Regularization (λ)	0.01
Batch Size	10
Epochs	150

Table 4.9: Spatial Pooler Parameters for CAE small-NORB

Parameter	**Symbol**	**Value**
Total columns in region	m	2048
Number of synapses per column	q	500
Permanence increment	ϕ_+	0.005
Permanence decrement	ϕ_-	0.005
Synapses connection threshold	ρ_s	0.5
Column activation threshold	ρ_d	5
Active bits in SDR	ρ_c	64

4.2 Discussion

4.2.1 Uniqueness Metric for Semantic Similarity

Both the t-SNE and uniqueness matrix were used to visually illustrate SDRs and the high-dimensional space they exist inside. The implementation of these methods was used to address the question of what metrics exist for evaluating the semantic content of SDRs. This is an important problem for HTM researchers because of the distributed nature of the representation where individual bits mean nothing in isolation. Current HTM theory does not describe any regeneration of the input from an SDR, so the conversion from the SDRs to input space requires further research. Although the Spatial Pooler processes only feedforward spatial information, it is a continuous online learning algorithm and adapts to changes in the spatial data. This is implemented with Hebbian learning, and results in changes to the SDRs as the nature of the input to the Spatial Pooler changes. Therefore the relationships among the SDRs may change over time as the Spatial Pooler adapts to new data. The assignment of significant meaning to any particular SDR may be shortsighted. There is a strong need for understanding the content of SDRs, and this work focuses on the relationships between SDRs as a basis for content evaluation.

The uniqueness metric is a visualization tool that uses the spatial semantic relationships in SDRs to evaluate the content of the SDRs, which was primarily done to aid researchers in understanding how to build HTM systems. The uniqueness matrix contain strict numerical values, but they were omitted in the plots. This was done purposely to focus on the general relationships between classes rather than the absolute differences/similarity. There is no desirable similarity between class 1 and class 2, class 2 and class 5, class x and y, and so on. The focus of this metric is for researchers to understand what semantic relationships the Spatial Pooler learns among the data, and to provide a consistent measure of the semantics. While the t-SNE plots are help-

ful in spatially illustrating the existence of clusters, they have inconsistent results and many tunable hyperparameters (perplexity, learning rate, exaggeration, theta). The t-SNE algorithm requires some prior knowledge of what the clusters might already look like in order to get accurate data clusters, and realistically this requires several runs of the t-SNE algorithm with different hyperparameters. The t-SNE plot also does not accurately visualize the distances between clusters effectively [27].

Instead the uniqueness matrix embraces the high-dimensionality and focuses on distances between the representations instead of imposing specific low dimensional clusters on the data. This approach is analogous to Representational Similarity Analysis [33], where the focus is on the distance between stimulus in high dimensional space.

The recent focus of machine learning algorithms has been that a good representation is one that is separable, and to some extent this value has been guiding the content evaluation of the Spatial Pooler SDRs. In contrast this work suggests that a good representation for HTM's SDRs is one that contains semantic information in addition to the sparsity, entropy, and noise robustess that are required for good SDRs. This ambiguous definition of good SDR content is explored in the next section by focusing on the encoder's effect on the semantic relationships.

4.2.2 Contractive Autoencoder as an HTM Encoder

The purpose of an HTM encoder is to define the semantic meaning among the input data, and to some extent this task is left up to the system designer. A major benefit of the CAE as an HTM encoder is that it establishes semantic meaning between images without hand crafted features. The contractive autoencoder was shown to extract meaningful spatial semantic information for the Spatial Pooler compared to the image thresholding encoder. It creates these representations based on the regularization term that encourages invariance or "contractions", so images that are spatially similar

have similar representations. However, the training paradigm of the CAE poses some questions concerning end-to-end compatibility with an HTM system. The CAE is trained with an optimization process to minimize the cost function, which may require several epochs to find the best optimization. **The optimization approach to training is a different paradigm of learning than the Hebbian learning utilized in the Spatial Pooler.** This will likely create training issues when a CAE and Spatial Pooler system is presented with novel data because there is difficulty training the CAE on new data without forgetting the existing representations. The encode novel data for the CAE and Spatial Pooler system requires retraining the CAE. In order to retrain the CAE new images would have to be presented with old images to prevent the CAE from forgetting how to encode existing images. In addition different datasets require different hyperparameters to achieve a saturated representation, whereas in HTM the standard hyperparameters work for a variety of datasets.

The CAE's training paradigm should not undermine it's ability to produce binary semantic representations for HTM, and there are benefits to having strong semantic representations between SDRs. Since the regularization term in the CAE produces contractions in directions of invariance, the representations learned will be locally invariant to small directions of change. At the extreme this was observed by the digit *1* (class 1) in MNIST because there was a high similarity among the representations according to the t-SNE plot and uniqueness matrix. This invariance to the input results in a larger variance in the similarity, μ, between the SDRs.

A larger variance in SDR similarity will have a crucial effect on the calculation of similarity between any two SDRs. It allows for a larger range of threshold values, θ, for establishing similarity or equivalence between any two SDRs after calculating their SDR overlap. However, the large amount of variance in similarity creates stronger similarity (larger μ) between SDRs of different classes. This creates stronger semantic

relationships between SDRs, so the indication of a match or equivalence between two SDRs may or may not occur depending on the threshold value.

4.2.3 Binding and Superposition

It was shown that the CAE creates strong semantic relationships, which correspondingly result in entangled clusters. For some applications it may be desirable to create separable representations. It was shown that the binding operation was able to untangle the relationships, and create separable clusters of SDRs according to class for the MNIST SDRs. However, in order to produce the separability each SDR was known to belong to a specific class, so it was bound with the appropriate cue SDR. **This demonstration was dependent on a supervised understanding of the data (i.e. each SDR had a class label), so an online or unsupervised approach to binding for separability would not be possible within the current HTM theory.** In addition the Spatial Pooler does not produce *maximally sparse* SDRs that are required for the binding of spare binary distributed vectors. This issue was overcome in the MNIST binding experiment by using the maximally sparse binary vector (cue SDR) as the index for the number of bits to shift. This limits the full functionality or capabilities of the binding operation that was described in [48]. The limitation means that only unbinding can be done with the cue SDR for this experiment. The remedy for this is likely to utilize local inhibition to create neighborhoods within the region that correspond to segments, and ensure that the result of each neighborhood produced a single column activation.

Binding and superposition were used to create an aggregated or robust representation for the small NORB subset. The binding of a specific location SDR with the sensory SDR will allow the recollection or extraction of features from the superimposed SDR. This means that the result of unbinding operation between a location SDR and the aggregated SDR will give only the sensory SDR that was bound with

that location SDR. This is effectively the basis for content addressable memory, and the binding operation is one of the methods for traversal of this memory organization. The benefits for storing data in the manner allows for a compact representation because the dimensionality does not increase, and information can be queried out through the binding/unbinding operation with cue SDRs. There are some remaining issues that will prevent the full functionality of content addressable memory for HTM's SDRs. Recollection will prove difficult because the SDRs generated by the Spatial Pooler are not maximally sparse, so cues generated by the Spatial Pooler cannot be used to unbind the representations. There is a substantial density increase that occurs as SDRs are superimposed, and when querying data out of superposition is unclear how to thin the recalled SDR to get the original SDR at the lower density. According to VSA theory, it is expected that the recalled SDR after unbinding will be noisy, and this can be remedied by using cleanup memory to retrieve the pristine version of the original SDR.

The creation of representations could be enhanced by the inclusion of Temporal Memory. The anomaly detection capability could be utilized to mark the creation of a new memory representation, which would allow for the unsupervised creation of complex SDRs. In other scenarios it could be used to unbind other memory representations, and potentially assign context to novel data based on stored representations. In this work the location signals/SDRs were bound with the Spatial Pooler SDRs, but it is unlikely this would occur in the neocortex. It is more likely that location signals are combined with the sensory information before it is processed by the Spatial Pooler [7]. However, this was done intentionally to represent that data of different domains can be combined together with the binding operation.

The specific combination of vector operations to combine and manipulate data is dependent on the architecture, or how the vector operations are utilized to create new representations. This work explored the binding and superposition operations

specifically for the small NORB dataset, but given additional SDRs the manipulation of this space may change. It may require binding or superposition of additional SDRs to get the desired representation, which is likely highly dependent on the application. In fact it is suggested that good architectural design is more important as these vector operations cannot be trained [14]. In summary the VSA vector operations are a set of tools that could be useful in creating robust relationships and untangling existing ones for the Sparse Distributed Representations in HTM, but the design and application requirements have a significant influence on how the representations are manipulated.

Chapter 5

Conclusion

The solution to the first research question was answered by examining the semantic relationships between Sparse Distributed Representations through a uniqueness matrix. A Contractive Autoencoder was used to address the second research for encoding the spatial information in images for HTM systems. The vector operations of Sparse Distributed Representations were expanded with a binding operation to address the third research question.

The Contractive Autoencoder was shown to creates spatial relationships for datasets where simple thresholding does not allow any significant relationships to be encoded. The binding operation for maximally sparse vectors illustrates how the Sparse Distributed Representations can be manipulated to form a separable representations, but the operation is limited due to the lack of maximally sparse representations produced by the Spatial Pooler.

5.1 Future Research Directions

The future research of the proposed questions can be extended in three directions. The uniqueness matrix function of illustrating the general semantic similarity among known data types is highly dependent on labeled data, and does not visualize the geometry of the sparse distributed representations. Because of the difficulties of visualization with t-SNE, further investigation into visualization and evaluation of

the content should look into representational similarity matrices would be beneficial. The contractive autoencoder requires stochastic gradient descent to optimize the cost function, which creates difficultly for continuous online learning systems where there is no assumptions about the structure of the input. A generic contractive autoencoder would be beneficial to alleviate this concern by training one contractive autoencoder to work for any image data. The binding operation can be made invertible for content addressable memory by exploring how local inhibition in the Spatial Pooler can be utilized to produce maximally sparse, Sparse Distributed Representations.

**All code and assets for this research is hosted at:*

– *https://kgcoe-git.rit.edu/lgb9267*

Bibliography

[1] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. The HTM Spatial PoolerA Neocortical Algorithm for Online Sparse Distributed Coding. *Frontiers in Computational Neuroscience*, 11(November):1–15, nov 2017.

[2] Peter Lennie. The cost of cortical computation. *Current biology*, 13(6):493–497, 2003.

[3] Jeremy Hsu. Ibm's new brain [news]. *IEEE spectrum*, 51(10):17–19, 2014.

[4] Jeff Hawkins and Sandra Blakeslee. *On intelligence:*. Times Books/Henry Holt, 2004.

[5] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10, 2016.

[6] Yuwei Cui, Subutar Ahmad, and Jeff Hawkins. Continuous online sequence learning with an unsupervised neural network model. *Neural Computation*, 2016.

[7] Jeff Hawkins, Subutai Ahmad, and Yuwei Cui. A Theory of How Columns in the Neocortex Enable Learning the Structure of the World. *Frontiers in Neural Circuits*, 11(October):1–18, oct 2017.

[8] James Mnatzaganian, Ernest Fokou, and Dhireesha Kudithipudi. A mathematical formalization of hierarchical temporal memorys spatial pooler. *Frontiers in Robotics and AI*, 3:81, 2017.

[9] Jeff Hawkins, Subutai Ahmad, and Donna Dubinsky. Hierarchical temporal memory including htm cortical learning algorithms. *Techical report, Numenta, Inc, Palto Alto http://www. numenta. com/htmoverview/education/HTM_CorticalLearningAlgorithms. pdf*, 2010.

[10] Subutai Ahmad and Jeff Hawkins. Properties of sparse distributed representations and their application to hierarchical temporal memory. *arXiv preprint arXiv:1503.07469*, 2015.

[11] Geoffrey E Hinton, James L McClelland, David E Rumelhart, et al. Distributed representations. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(3):77–109, 1986.

[12] Scott Purdy. Encoding Data for HTM Systems. *arXiv*, page 1602.05925, feb 2016.

[13] Ray Jackendoff. *Foundations of language: brain, meaning, grammar, and evolution*. Oxford University Press, 2003.

[14] Ross W Gayler. Vector symbolic architectures answer jackendoff's challenges for cognitive neuroscience. *arXiv preprint cs/0412059*, 2004.

[15] Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

[16] Peter Blouw, Eugene Solodkin, Paul Thagard, and Chris Eliasmith. Concepts as Semantic Pointers: A Framework and Computational Model. *Cognitive Science*, 40(5):1128–1162, jul 2016.

[17] Christoph von der Malsburg. Binding in models of perception and brain function. *Current Opinion in Neurobiology*, 5(4):520–526, aug 1995.

[18] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[19] Dmitri A Rachkovskij and Ernst M Kussul. Binding and normalization of binary sparse distributed representations by context-dependent thinning. *Neural Computation*, 13(2):411–452, 2001.

[20] T. A. Plate. Holographic reduced representations. *IEEE Transactions on Neural Networks*, 6(3):623–641, May 1995.

[21] Ross W Gayler and Roger Wales. Connections, binding, unification and analogical promiscuity. 1998.

[22] Pentti Kanerva. *Sparse distributed memory and related models*, volume 92. Citeseer, 1992.

[23] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[24] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[25] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 833–840, 2011.

[26] Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot. Higher order contractive auto-encoder. *Machine Learning and Knowledge Discovery in Databases*, pages 645–660, 2011.

[27] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[28] Martin Wattenberg, Fernanda Vigas, and Ian Johnson. How to use t-sne effectively. *Distill*, 2016.

[29] Samuel A. Nastase and James V. Haxby. 3.09 - structural basis of semantic memory. In John H. Byrne, editor, *Learning and Memory: A Comprehensive Reference (Second Edition)*, pages 133 – 151. Academic Press, Oxford, second edition edition, 2017.

[30] Endel Tulving. Episodic and semantic memory: Where should we go from here? *Behavioral and Brain Sciences*, 9(3):573–577, 1986.

[31] Morris Moscovitch, Roberto Cabeza, Gordon Winocur, and Lynn Nadel. Episodic memory and beyond: the hippocampus and neocortex in transformation. *Annual review of psychology*, 67:105–134, 2016.

[32] EricR. Kandel, Yadin Dudai, and MarkR. Mayford. The molecular and systems biology of memory. *Cell*, 157(1):163 – 186, 2014.

[33] Nikolaus Kriegeskorte, Marieke Mur, and Peter Bandettini. Representational similarity analysis–connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2, 2008.

[34] Barry J Devereux, Alex Clarke, Andreas Marouchos, and Lorraine K Tyler. Representational similarity analysis reveals commonalities and differences in the semantic processing of words and objects. *Journal of Neuroscience*, 33(48):18906–18916, 2013.

[35] Seyed-Mahdi Khaligh-Razavi, Linda Henriksson, Kendrick Kay, and Nikolaus Kriegeskorte. Fixed versus mixed RSA: Explaining visual representations by fixed and mixed feature sets from shallow and deep computational models. *Journal of Mathematical Psychology*, 76(April):184–197, feb 2017.

[36] James W. Mnatzaganian. A mathematical formalization of hierarchical temporal memory's spatial pooler for use in machine learning. Master's thesis, Rochester Institute of Technology, 1 Lomb Memorial Drive, Rochester, NY, 4 2016.

[37] Leland McInnes and John Healy. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. pages 1–18, feb 2018.

[38] Junichi Chikazoe, Daniel H. Lee, Nikolaus Kriegeskorte, and Adam K. Anderson. Population coding of affect across stimuli, modalities and individuals. *Nature Neuroscience*, 17(8):1114–1122, jun 2014.

[39] Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot. Higher Order Contractive Auto-Encoder. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6912 LNAI, pages 645–660. 2011.

[40] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

[41] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[42] Yann LeCun, Fu Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–104. IEEE, 2004.

[43] James W. Antony, Catarina S. Ferreira, Kenneth A. Norman, and Maria Wimber. Retrieval as a Fast Route to Memory Consolidation. *Trends in Cognitive Sciences*, 21(8):573–576, aug 2017.

[44] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-Inspired Artificial Intelligence. *Neuron*, 95(2):245–258, jul 2017.

[45] Gabriel Makdah. Modeling the Mind: A brief review. pages 1–52, 2015.

[46] Daniel E. Padilla and Mark D. McDonnell. A neurobiologically plausible vector symbolic architecture. *Proceedings - 2014 IEEE International Conference on Semantic Computing, ICSC 2014*, pages 242–245, 2014.

[47] Terrence C Stewart, Trevor Bekolay, and Chris Eliasmith. Neural representations of compositional structures: Representing and manipulating vector spaces with spiking neurons. *Connection Science*, 23(2):145–153, 2011.

[48] Mika Laiho, Jussi H Poikonen, Pentti Kanerva, and Eero Lehtonen. High-dimensional computing with sparse vectors. In *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, pages 1–4. IEEE, 2015.